Universidad Carlos III de Madrid

Proyecto de Fin de Carrera

Descubrimiento y geolocalización mediante redes Wi-Fi

Autor: Carlos Pérez Blanco
Tutor: Yago Sáez Achaerandio
Co-Tutor: Gustavo Recio Isasi
Fecha: 13 de julio de 2010

Agradecimientos

La presentación y defensa del Proyecto de Fin de Carrera representa el cierre no únicamente de varios meses de duro y constante trabajo, sino también de un ciclo de nuestras vidas, posiblemente uno de los más importantes. Han sido más de seis años de trabajo en la Universidad Carlos III de Madrid del cual, especialmente en los últimos años, me siento orgulloso.

Empezaré por agradecer a mis padres en especial, y toda mi familia en general, haberme dado la posibilidad de encontrarme donde me encuentro y haberme apoyado siempre en todas mis decisiones. Sin ellos nada de esto habría sido posible.

También quiero dar las gracias a todos los compañeros que he tenido la suerte de conocer a lo largo de estos años, tanto en la Universidad Carlos III como durante mi año como estudiantes Erasmus en la Tampere University of Technology (Tampere, Finlandia) y con los que he compratido muchos momentos, tanto dentro como fuera de la Universidad. En especial a Lucas, mi eterno compañero de prácticas; Alberto, compañero de aventuras en Tampere; Jorge, Ismael y Fran de la UC3M y Héctor de TUT.

Por último, me gustaría dar las gracias a Yago, mi tutor, por su orientación, sus consejos y su apoyo durante estos meses, así como a Emilio, Moisés y Gustavo, por su excelente disponibilidad y toda la ayuda que me han prestado. Realmente ha sido un placer tratar con todos, y gracias a ellos el trabajo ha sido mucho más sencillo.

Resumen

En la actualidad la gran mayoría de dispositivos móviles disponen de diversos sistemas de localización, siendo el GPS el más habitual. Gracias a ello nuestro dispositivo conoce en cualquier momento la posición en la que nos encontramos, dando pie a un extenso abanico de posibilidades. Posiblemente la más popular sea la navegación, pero conocer nuestra localización tiene muchas otras aplicaciones. Por ejemplo, podemos crear una aplicación de chat en la que las personas con las que podemos *chatear*, son aquellas que se encuentran más cerca físicamente de nosotros.

Este proyecto busca analizar y comprender los fundamentos de la geolocalización mediante redes Wi-Fi, conociendo sus ventajas y desventajas frente a otras alternativas como el propio GPS. Se desarrollará una aplicación para dispositivos móviles con Android instalado que sea capaz de determinar la posición geográfica de un usuario mediante la información de las redes Wi-Fi recibida por el dispositivo móvil.

Por otra parte, el desarrollo de la aplicación específica para Android servirá de igual modo para conocer las características y los fundamentos básicos de programación para el nuevo sistema operativo de Google.

Abstract

Nowadays most of mobile devices feature different positioning capabilities, being GPS capability the most frequent, so our device knows where we are at any point in time. This feature gives cause to a great potential. Possibly the most common use is mapping and navigation, but knowing our location has a lot of different uses. For instance, we could create a chat application where the people we could chat with are those you are nearest.

This projects aims to analyze and understand the basics of geopositioning through Wi-Fi networks, knowing its advantages and disadvantages over other alternatives like the actual GPS. An application will be developed for Android devices, which will be able to determine the geographic position of the user based on the information about the Wi-Fi networks received by the device.

On the other hand, developing an application specifically for Android devices will also make possible to get to know the characteristics and fundamentals of programming for the new operating system by Google.

Índice general

Índice de figuras

Índice de código

Capítulo 1

Introducción

En este capítulo se hará una pequeña introducción al proyecto, exponiendo su motivación y los objetivos perseguidos, así como una breve descripción de los contenidos de la presente memoria.

1.1. Motivación

En la actualidad, las nuevas tecnologías forman una parte esencial de nuestras vidas, y parecen avanzar sin descanso a un ritmo vertiginoso. Desde el descubrimiento de la informática a mediados del sigo XX, dos grandes inventos destacan por encima de los demás: el **ordenador** e **Internet**, conceptos que hoy en día van de la mano y prácticamente el uno no puede entenderse sin el otro. Por otra parte, a finales del siglo pasado, un invento más antiguo, la telefonía, se aprovechó de los avances en el campo de la informática, surgiendo la **telefonía móvil**. Todas estas tecnologías son actualmente conocidas y usadas en el día a día por gran parte de la población mundial.

Poco a poco ambos campos han ido acercándose uno al otro, a medida que se han producido avances en la reducción del tamaño de los componentes y en el abaratamiento de los costes, dando lugar a los **dispositivos móviles**. Si bien es un concepto general, un dispositivo móvil se puede definir como un dispositivo de reducido tamaño con la capacidad de procesar, recibir y enviar información, que puede ser usado en diferentes entornos. Entre los dispositivos móviles podemos encontrar PDAs, teléfonos móviles o los nuevos ordenadores *ultraportables*.

Las características de este tipo de dispositivos dan lugar a un amplio abanico de posibilidades. Entre ellas podemos destacar el gran número de servicios ofrecidos al usuario basados en su localización. Y es que un dispositivo móvil, y por lo tanto el usuario del mismo, pueden estar localizados en todo momento, y actuar en función del lugar donde se encuentre.

Las aplicaciones son prácticamente infinitas. Desde posibilidades sencillas, como poder incluir en una fotografía tomada con nuestro teléfono móvil información sobre el lugar desde el que se tomó, hasta otras más complejas, como una aplicación de chat en la que las personas con las que podemos *chatear*, son aquellas que se encuentran más cerca físicamente de nosotros, o la posibilidad de descargar la información sobre la Catedral de la Almudena en Madrid mientras nos disponemos a entrar en ella. Incluso en materia de seguridad, la posibilidad de localizar un dispositivo puede ayudar a recuperar material robado, o detectar dispositivos no autorizados que pudieran suponer algún tipo de riesgo.

Todo ello ha dado lugar al concepto de *geolocalización*, geoposicionamiento o georreferencia.

¿Qué es la **geolocalización**? El concepto es tan novedoso que la Real Academia Española aún no incluye una definición para algún termino similar, aunque se puede concluir que se refiere a la determinación de la posición geográfica de una persona o un dispositivo.

¿Cómo se logra geolocalizar un dispositivo? Existen diferentes sistemas de posicionamiento, tanto a nivel global como local. Los sistemas de posicionamiento globales, como el GPS, permiten localizar un dispositivo en cualquier parte del mundo, ya que hacen uso de una tecnología con cobertura global. Por contra, los sistemas de posicionamiento locales hacen uso de tecnología con cobertura local, como antenas de telefonía o redes Wi-Fi. Siempre que un dispositivo móvil cuente con las características y el *hardware* necesarios, es posible utilizar cualquiera de estas opciones a nuestro favor.

Sin embargo, no todos los sistemas de posicionamiento resultan adecuados en cualquier circunstancia. Algunos de ellos, como el GPS o la triangulación mediante celdas de telefonía, resultan imprecisos, lentos o inadecuados bajo determinadas circunstancias, como en interiores de edificios o zonas con baja cobertura. Es por ello que un sistema de geoposicionamiento mediante redes Wi-Fi, aprovechando los miles de puntos de acceso Wi-Fi situados en cualquier área poblada, puede ayudar a proporcionar información precisa de localización tanto en interiores como en zonas urbanas.

Por último, otra de las premisas impuestas para realizar el proyecto fue utilizar la opción de **Android** como plataforma para el desarrollo del sistema, por diferentes motivos. El nuevo sistema operativo de Google, lanzado el 21 de Octubre de 2008, es un sistema operativo *open source* y gratuito, con las ventajas y sobre todo las libertades que estas características traen consigo. El software *open source* está en auge y Android, con menos de dos años de vida, ya es un serio competidor cuya cuota de mercado no deja de crecer.

1.2. Objetivos

El objetivo principal de este proyecto es el de diseñar un sistema para Android que permita descubrir señales — redes — Wi-Fi y posicionarlas en función de sus intensidades. Una vez almacenada su posición, se desarrollará una aplicación que utilice dichos datos para determinar la posición geográfica del usuario.

Los datos de las posiciones de las redes Wi-Fi deben estar almacenacenados en un servidor, donde puedan ser actualizados en cualquier momento por un usuario cualquiera del sistema.

Este objetivo se puede descomponer en dos grandes bloques, por un lado la recolección de datos de redes inalámbricas y por otro, la utilización de los mismos para la geolocalización del usuario. Estos dos bloques, además, se corresponden con cada una las dos aplicaciones que se desarrollarán.

La primera de las aplicaciones estará enfocada a la recolección de datos de redes Wi-Fi. Se trata de una herramienta colaborativa, en la que cualquier usuario puede recoger datos acerca de las posiciones geográficas de los puntos de acceso detectados, para después subirlos si lo desea a un servidor, donde se almacenarán para su posterior consulta. Dada la importancia que tendrán estos datos para la fiabilidad de la segunda parte del proyecto resultará vital conseguir los mejores resultados posibles en cuanto a precisión.

La segunda aplicación, por su parte, es la que hará uso de los datos recogidos previamente para determinar la posición geográfica del usuario. La aplicación será capaz de ofrecer unas coordenadas basándose únicamente en la información de las redes Wi-Fi al alcance del dispositivo móvil donde se ejecute la aplicación. Como se ha comentado anteriormente, la fiabilidad de la aplicación estará condicionada por la de los datos usados.

Al margen del objetivo principal, existe una serie de objetivos paralelos que será necesario alcanzar a medida que se avance en el desarrollo del proyecto, los cuales se enumeran a continuación:

- Conocer las técnicas de posicionamiento de puntos de acceso resulta de gran importancia para el proyecto, pues se debe conseguir la mayor precisión posible en la localización de las redes descubiertas.
- Será necesario conocer y estudiar los principales rasgos característicos de Android, su arquitectura y sus componentes, así como conocer también el entorno sobre el que se desarrollarán las aplicaciones.

1.3. Contenido de la memoria

Esta memoria se compone de seís capítulos y un anexo, cuyo contenido se describe brevemente a continuación:

En el capítulo 1, *Introducción*, se expone la motivación del proyecto y los objetivos que se han perseguido durante su desarrollo. Igualmente, se ofrece una visión general de los contenidos de la memoria.

El capítulo 2, *Estado del Arte*, ofrece una descripción general de varios de los aspectos relacionados con el presente proyecto. En concreto, se profundizará en las técnicas de recolección de datos y mapeo de redes inalámbricas, así como en los sistemas de geolocalización basados en redes Wi-Fi.

En el capítulo 3, *Recogida de datos y mapeo de redes Wi-Fi: La aplicación Warwalk*, se analizará la aplicación dedicada al descubrimiento, recolección de datos y posicionamiento de redes Wi-Fi, a partir de la cual se ha desarrollado una versión modificada adaptada a los objetivos de este proyecto. Se debatirá la necesidad de realizar estas modificaciones y se detallarán los cambios aplicados.

En el capítulo 4, *Geolocalización mediante redes Wi-Fi: La aplicación WPS*, se detalla el desarrollo completo de la aplicación para la geolocalización del usuario mediante redes Wi-Fi. Partiendo del diseño, requisitos y funcionalidad requerida, hasta la implementación de todos sus componentes.

En el capítulo 5, *Experimentación y Resultados*, se describen las diferentes pruebas y experimentos realizados con la aplicación, una vez finalizado su desarrollo, con el fin de asegurar su correcto funcionamiento y obtener unos resultados con los que poder analizar el éxito del proyecto.

El capítulo 6, *Conclusiones y Trabajos Futuros*, analiza los resultados obtenidos a través de los experimentos anteriores y los contrasta con los objetivos planteados en un primer momento para el proyecto, extrayendo las conclusiones oportunas. Por otra parte se proponen líneas de trabajo futuras, así como se ofrece una pequeña visión de lo que ha supuesto el desarrollo del proyecto.

El *Anexo A. Instalación de las aplicaciones* detalla la instalación de las aplicaciones en un dispositivo Android compatible.

Capítulo 2

Estado del Arte

A lo largo de este capítulo se presenta una revisión del estado del arte en lo referente al descubrimiento y la geolocalización mediante redes Wi-Fi. Se expondrá su origen y evolución, se definirán los conceptos básicos con los que el lector debe familiarizarse, y se describirán y analizarán diferentes aplicaciones y trabajos relacionados que servirán como base para el futuro desarrollo del proyecto.

Para ayudar al lector en la lectura y comprensión del capítulo, éste, al igual que el objetivo principal de este proyecto, se ha estructurado en dos secciones. Para cada una de estas secciones se ha incluido además una subsección en la que se debatirá el estado de la cuestión para dispositivos Android.

La primera sección tratará los temas relacionados con la recogida de datos y mapeo de redes Wi-Fi y la manera en que se mapean, es decir, se traslada a un mapa su ubicación. Se introduce el concepto de *warXing* y se analizarán varias de las aplicaciones disponibles para este fin, así como las técnicas que utilizan.

La segunda sección tratará los diferentes sistemas de geolocalización mediante redes Wi-Fi disponibles y sus características.

Esta división será la tónica habitual a lo largo del presente documento, pues se trata de dos bloques completamente independientes entre sí, si bien uno de ellos — los sistemas de geolocalización mediante redes Wi-Fi — necesita del paso previo — la recogida de datos — para poder realizarse, por lo que se ha optado por mantener ambos bloques separados en todo momento.

2.1. Recogida de datos y mapeo de redes Wi-Fi: *WarXing*

2.1.1. Introducción

La detección de ordenadores o redes inalámbricas accesibles públicamente se denomina en inglés *warXing*. Este término deriva de la técnica utilizada por uno

de los personajes de la película *WarGames*, en la que se usaba un ordenador para marcar números de teléfono consecutivos de manera automática, con la esperanza de encontrar algún módem activo. A esta técnica se la denominaba en la película *wardialing*, en base al propio título de la misma [1]. El concepto, por tanto, sigue siendo el mismo, pero aplicado a una tecnología más actual: las redes inalámbricas.

La X del término *warXing* puede sustituirse para especificar en mayor medida la actividad realizada, dando lugar, entre otros, a los siguientes términos:

- *Wardriving*: detectar redes Wi-Fi desde un vehículo equipado con algún tipo de dispositivo Wi-Fi a la vez que se conduce.
- *Warwalking*: buscar redes inalámbricas andando alrededor de una zona.
- *Warchalking*: marcar la localización de una red inalámbrica activa mediante marcas de tiza en las aceras.
- *Wardialing*: detectar ordenadores conectados a la red telefónica marcando todos los números de teléfono de un área.
- *Warflying*: detectar redes Wi-Fi desde un avión.

De todas las formas presentadas anteriormente la más habitual sin lugar a dudas es *wardriving*, por lo que de aquí en adelante se utilizará este término para referirse indistintamente a cualquiera de las actividades presentadas.

La idea de conducir alrededor de una cierta zona recogiendo datos de redes inalámbricas de manera automática se le puede atribuir a Peter Shipley, quien durante el otoño del año 2000 realizó durante 18 meses un estudio acerca de la falta de seguridad de las redes inalámbricas en Berkeley, California.

Si bien la primera impresión puede llevar a pensar en el *wardriving* como una actividad minoritaria y sin demasiada utilidad, lo cierto es que existen muchas comunidades de personas que tienen como *hobby* recorrer ciudades enteras recopilando datos de todas las redes inalámbricas a su alcance.

Pero, ¿cuál es exactamente el objetivo del *wardriving*? Como se ha comentado puede tratarse de un simple *hobby*, pero sin embargo la finalidad última es proporcionar información al público sobre la vulnerabilidad y la falta de seguridad en las configuraciones de los puntos de acceso inalámbricos, especialmente con las configuraciones *out of the box* — de fábrica —.

Para poder llevar a cabo la actividad, es necesario el uso de hardware y software específico. En lo referente al hardware, lógicamente en todos los casos es necesario contar con algún tipo de dispositivo equipado con Wi-Fi, como puede ser un ordenador portátil o una PDA para poder detectar las redes. Habitualmente se utilizan tarjetas wireless de gama alta y antenas externas —direccionales u omnidireccionales—, para mejorar la cobertura y la potencia de las señales recibidas, y con ello los resultados obtenidos. También es frecuente el uso de un dispositivo GPS para determinar y almacenar las coordenadas de los puntos de acceso detectados.

Por otro lado también debe usarse, al menos, un software específico para *wardrive*. Este software generalmente tiene la capacidad de descubrir redes automáticamente, si bien puede contar igualmente con otras características como la monitorización de las señales de radio recibidas y la posibilidad de plasmar en un mapa las posiciones de las redes descubiertas, en caso de que se hayan guardado sus coordenadas.

Es importante destacar que estas actividades tienen como único propósito la recopilación de información de los diferentes puntos de acceso inalámbricos — WAP, del inglés *Wireless Access Point* —. En ningún momento se pretende hacer uso de los servicios ofrecidos por dichos puntos de acceso, actividad conocida como *piggybacking* y que puede ser ilegal en función de las leyes vigentes.

Llegados a este punto y una vez conocidos los conceptos básicos del *wardriving*, podemos pasar a analizar algunas de las muchas aplicaciones disponibles en este campo. Lo más habitual hasta el momento es encontrar software disponible para ordenador, pues ha sido el medio habitual para la recolección de datos. Sin embargo, con el auge de los dispositivos móviles, cada vez son más los desarrolladores que se interesan en crear aplicaciones para ellos.

2.1.2. Kismet

Kismet [2] es uno de los mejores y más usados programas a la hora de recoger datos de redes Wi-Fi. Se trata de un detector, rastreador y detector de intrusiones de redes inalámbricas 802.11. Es capaz de rastrear tráfico en redes 802.11b, 802.11a, 802.11g y 802.11n. Está distribuido bajo licencia GNU General Public License y por tanto es software libre. Funciona únicamente en sistemas Linux, si bien existe una versión *hermana* para Macintosh llamada KisMAC [3], con la que sin embargo guarda ciertas diferencias.

Kismet, al contrario que otros programas, funciona de manera pasiva, es decir, es capaz de detectar las redes sin enviar ningún paquete. Respecto a la detección de intrusiones, el programa tiene capacidad para detectar la presencia de otro software activo como NetStumbler y otros tipos de ataques a la red.

Kismet es igualmente capaz de integrarse con un dispositivo GPS para proporcionar coordenadas para las redes detectadas. Estas coordenadas pueden ser exportadas a formato XML para su uso en otras herramientas, como Kismap.

La aproximación que toma Kismet a la hora de posicionar las redes Wi-Fi es sencilla, y coincide con la que será implementada en la aplicación a desarrollar. Kismet asume que «no puede conocer las coordenadas de un punto de acceso, sino únicamente las coordenadas desde donde se detectó su señal». Para obtener una medida más fiable es necesario rodear la zona donde se cree que está el punto de acceso. Kismet calcula constantemente las medias de las diferentes posiciones. Según la propia documentación en la web, este método «no es increíblemente preciso, debido al promedio y a la imprecisión en las operaciones en punto flotante» (*sic*).

2.1.3. NetStumbler

De entre el software disponible para Windows, NetStumbler [4] es el más utilizado. De hecho, es tal su popularidad que en ocasiones se usa el término *netstumbling* o *stumbling* en lugar del habitual *wardriving*. NetStumbler es una herramienta para el descubimiento y análisis de redes inalámbricas. Ha sido desarrollada por Marius Milner y la primera versión fue publicada a comienzos del año 2001. La aplicación es capaz de detectar redes basadas en 802.11b, 802.11g y 802.11a.

NetStumbler, a diferencia de Kismet, es un escáner activo. Ya se ha visto cómo un escáner pasivo recoge los datos sin necesidad de enviar ninguna señal. Por el contrario, un escáner activo como NetStumbler envía cada segundo una señal de sondeo, llamada *Probe Request*, que forma parte del propio estándar 802.11. Básicamente, la función de esta señal es la de *preguntar* si existe alguna red. Si alguna red responde, lo hará incluyendo entre otros datos, su BSSID y SSID.

Entre las numerosas opciones de configuración que ofrece el programa, destaca la posibilidad de configurar un dispositivo GPS para almacenar las coordenadas de los puntos de acceso. Como en el caso de Kismet, NetStumbler almacena las coordenadas desde las que una red inalámbrica fue detectada, sin realizar ningún cálculo adicional.

Por otra parte, los datos recogidos por NetStumbler pueden almacenarse en ficheros con formatio `.ns1` que son aceptados por numerosas herramientas para el mapeo de los puntos de acceso.

2.1.4. WiGLE

De entre las herramientas para el mapeo de las redes Wi-Fi, hoy en día se puede destacar especialmente WiGLE. WiGLE [5], del inglés *Wireless Geographic Logging Engine*, es un sitio Web dedicado a la recolección de datos de redes Wi-Fi de todo el mundo, almacenándolos en una base de datos centralizada y ofreciendo acceso a la misma a través de diferente software.

Cualquiera puede registrarse en el sitio y comenzar a subir datos de las redes capturadas, de manera que todos los datos recogidos están asociados a un determinado usuario. El registro también es necesario para poder realizar consultas en la base de datos o descargar el software disponible.

A través del propio sitio Web es posible consultar los datos almacenados, pudiendo obtener así información de millones de redes Wi-Fi en todo el mundo. Es posible buscar redes a través de una dirección postal (sólo para Estados Unidos), un rango de coordenadas, cierto SSID o incluso el propio BSSID o MAC del punto de acceso, entre otros.

A fecha de 21 de Junio de 2010 WiGLE, que lleva en funcionamiento desde Septiembre de 2001, cuenta con cerca de 23 millones de redes Wi-Fi almacenadas en su base de datos. Gran parte de esta cantidad se encuentra en Norteamérica, si

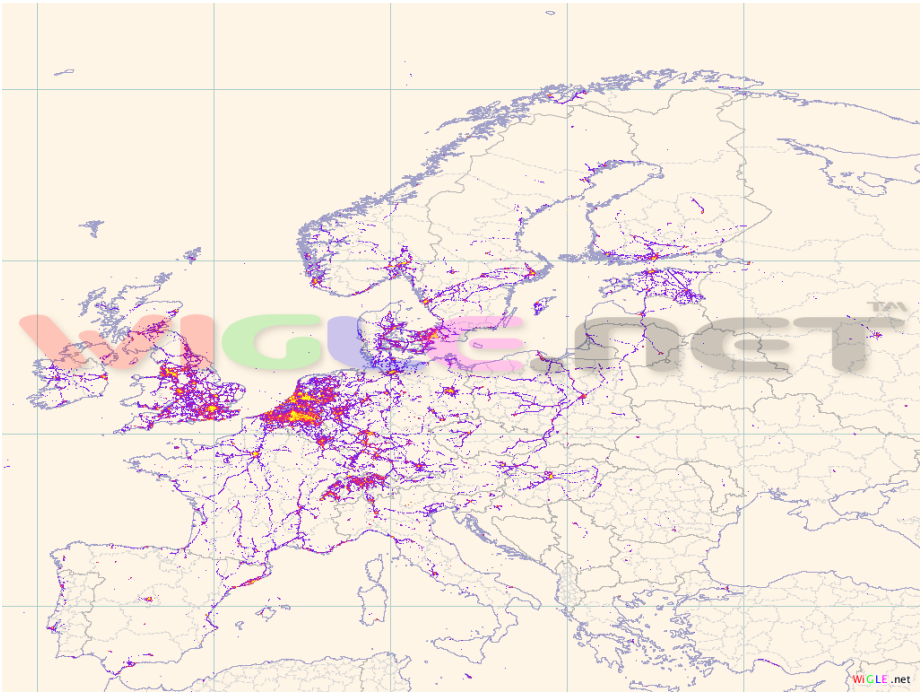

Figura 2.1: Mapa de redes Wi-Fi en Europa

bien en Europa se sube cada vez más información, como se puede observar en la figura 2.1. Cabe destacar que es posible solicitar la retirada de una determinada red Wi-Fi de la base de datos, en caso de que el propietario en cuestión no desee que aparezca en ella.

Respecto al software disponible, se trata de diferentes clientes para acceder a la base de datos, siendo una alternativa al sitio Web. Existen clientes Java — JiGLE —, Delphi — DiGLE —, y Mac OSX — TiNGLE —. El último de ellos ya no recibe soporte alguno.

Dado que WiGLE se dedica simplemente a centralizar los datos de las redes Wi-Fi, éstos han de obtenerse utilizando otro software diferente, para después subir el fichero generado por el mismo. WiGLE acepta la mayoría de formatos más usuales de aplicaciones como Kismet, NetStumbler, sus versiones para Macintosh, KisMAC y MacStumbler, o software para Android como G-Mon, y más recientemente Wardrive-Android o el propio WigleWiFi, entre otros.

Resulta especialmente interesante para el proyecto la manera en que se mapean los puntos de acceso a partir de la información obtenida. Según el propio FAQ (del inglés *Frequently Asked Questions*, Preguntas Frecuentes) del sitio [6], la triangulación es en realidad una media de las coordenadas recolectadas ponderadas por el cuadrado del nivel de señal, asumiendo que el nivel de potencia de la señal será inversamente proporcional al cuadrado de la distancia — al punto de acceso —. Esta aproximación fue la que finalmente se reproduciría a la hora de determinar la posición de las redes Wi-Fi para su uso posterior en el

proyecto.

Además, recientemente WiGLE ha lanzado su propio software para Android, con el que poder subir datos de manera sencilla desde el propio dispositivo móvil. Se hablará más en detalle de esta aplicación en la siguiente sección.

2.1.5. *Wardriving* en Android

Dado que el sistema se va a implementar para el nuevo sistema operativo de Google, Android, resulta oportuno hacer un repaso por algunas de las aplicaciones publicadas en Android Market [7] orientadas a la recolección de datos de redes Wi-Fi, mencionando las características básicas de cada una de ellas. Si bien la información disponible resulta en ocasiones escasa, limitándose incluso a la descripción ofrecida en el propio Android Market, se ha intentado recopilar tantos datos como ha sido posible.

El hecho de que, pese a la relativa novedad y el poco tiempo que Android lleva en el mercado, no sólo exista una gran cantidad de software disponible, sino que además éste cuente con miles de descargas, no hace sino confirmar el gran número de aficionados a actividades de tipo *wardriving*.

A favor de Android juega el hecho de disponer en un mismo dispositivo de reducido tamaño de todo lo necesario para la práctica del *wardriving*, haciendo así que resulte mucho más cómodo y sencillo recoger información. Por otro lado, si bien otros dispositivos con las características de hardware necesarias, como iPhone, podrían contar con software similar, la política restrictiva de Apple en cuanto a la publicación de aplicaciones en su App Store [8] [9] [10] [11] ha traido consigo el desinterés de los desarrolladores en esta plataforma, en contraste a lo ocurrido en Android, opción hacia la que se han más visto atraidos.

G-Mon

G-Mon [12] es un escáner para *wardriving* en Android. Está basada en la herramienta PyNetMony [13] para Symbian. La aplicación escanea las redes Wi-Fi dentro del rango del dispositivo y guarda sus datos junto con las coordenadas GPS en un fichero.

Dispone de un modo mapa, en el que poder visualizar la posición de las redes almacenadas. Además es posible exportar el fichero generado por la aplicación a diferentes formatos como .kml — para su visualización en Google Maps o Google Earth —, o .csv — valores separados por comas —.

La aplicación se encuentra publicada en Android Market, donde se puede descargar gratuitamente y cuenta con más de 10000 descargas de usuarios.

G-Mon ha sido una de las alternativas más valoradas a la hora de reutilizar alguna de las aplicaciones disponibles para el descubrimiento de redes Wi-Fi. Sin embargo la imposibilidad de acceder al código fuente de la misma, junto con la poca información disponible, finalmente resultó en que fuese desestimada.

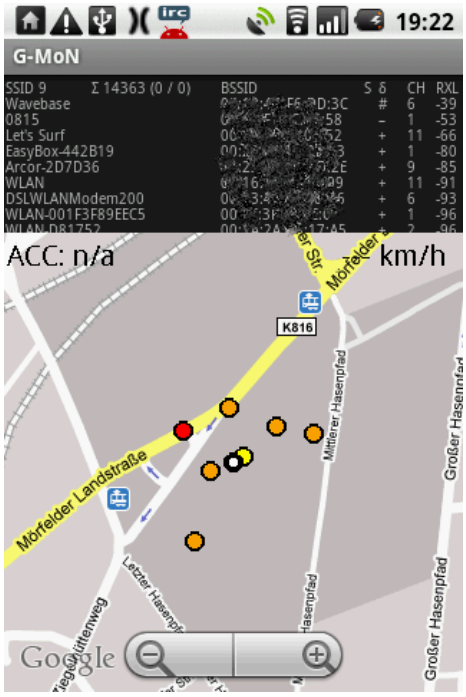

Figura 2.2: G-Mon para dispositivos Android

Wifi Tracker

Otra de las aplicaciones disponibles es Wifi Tracker [14], desarrollada por Ian Hawkins [15]. Este software pasa por ser uno de los más completos atendiendo a la funcionalidad ofrecida.

Además de recabar diferente información acerca de las redes inalámbricas al alcance del dispositivo — dirección MAC, seguridad y tipo de encriptación, coordenadas — y poder exportar los resultados a un fichero `kml` para su posterior visualización con Google Earth, la aplicación suma otro tipo de opciones menos habituales, como son la posibilidad de subir la información recabada a los servidores de WiGLE o incluso a un servidor propio, para lo que se ofrece la posibilidad de introducir su dirección, y el usuario y contraseña requeridos.

Lamentablemente el software no es libre, por lo que no es posible acceder a su código fuente. Tampoco existe ningún tipo de documentación disponible con la que poder entrar en detalles acerca de la implementación ni de las técnicas utilizadas.

En la figura 2.3 se pueden observar dos capturas de pantalla de la aplicación. La primera de ellas representa la vista habitual de la misma, mientras que la segunda muestra parte del menú de opciones.

Como es habitual, la aplicación está publicada en Android Market, sin embargo en este caso no se pone a disposición de los usuarios de manera gratuita,

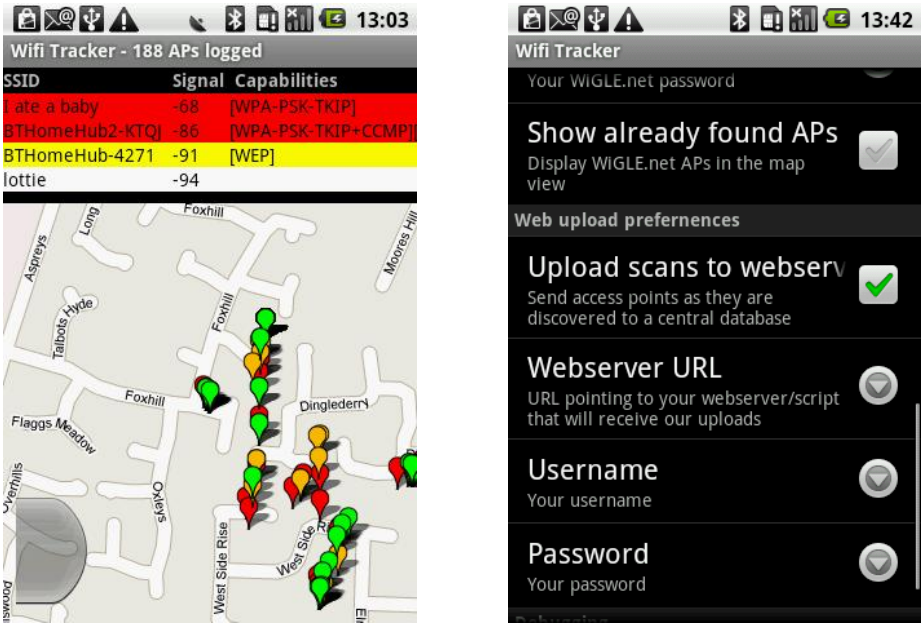

Figura 2.3: Capturas de pantalla de la aplicación Wifi Tracker

sino que puede adquirirse por un precio de £0.99. La aplicación, a fecha de la redacción de esta memoria, ha sido descargada entre 1000 y 5000 ocasiones.

WigleWiFi

Recientemente WiGLE ha lanzado su propio software para wardriving para dispositivos Android. Con esta aplicación, llamada WigleWiFi, podemos descubrir redes Wi-Fi y subir los datos a los servidores de Wigle desde cualquier dispositivo móvil con Android instalado. La figura 2.4 muestra el aspecto de la aplicación. En el momento de escribir esta memoria, la última versión disponible era la versión 1.8, que puede descargarse tanto de Android Market como desde el propio sitio Web.

Al igual que el resto de aplicaciones, WigleWiFi está disponible y en esta ocasión de manera gratuita, a través de Android Market, donde supera las 5000 descargas.

Wardrive-Android

Wardrive-Android [32] o simplemente Wardrive es una aplicación de *wardrive* desarrollada por Raffaele Ragni para dispositivos Android.

Esta aplicación ha sido utilizada como base del trabajo para la recogida de datos de redes Wi-Fi, por lo que se remite al lector al capítulo 3, donde

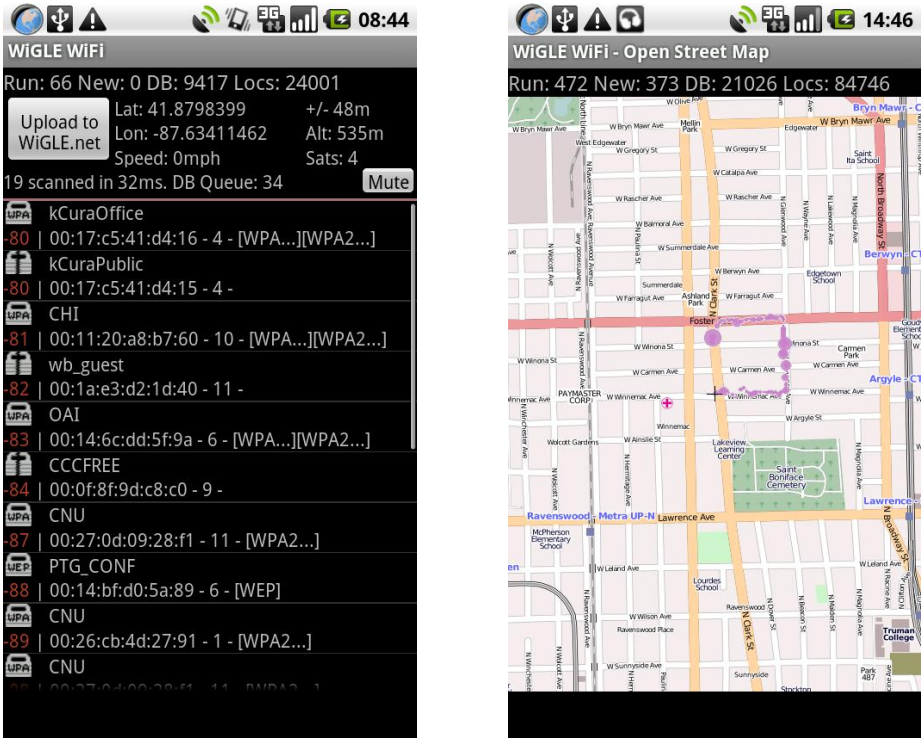

Figura 2.4: WigleWiFi para dispositivos Android

podrá encontrar una descripción completa de sus características y un análisis detallado de la funcionalidad ofrecida.

2.2. Sistemas de geolocalización mediante redes Wi-Fi

Los datos recogidos haciendo uso de las técnicas de *wardriving*, hacen posible la existencia de diferentes sistemas de geolocalización mediante redes Wi-Fi, que surgieron en gran medida como alternativa al uso del GPS.

En esta sección se analizarán algunos de los sistemas de geolocalización mediante redes Wi-Fi disponibles actualmente.

2.2.1. Skyhook Wireless

Una de las pioneras y más exitosas empresas en desarrollar un sistema de geolocalización mediante redes Wi-Fi fue Skyhook Wireless [17]. Se fundó en el año 2003 en Boston, Estados Unidos, y entre sus actividades destacan los servicios basados en localización, así como el desarrollo de tecnologías de posicionamiento.

El sistema de geolocalización de Skyhook Wireless, denominado *Wi-Fi Positioning System* (WPS), hace uso de las direcciones MAC de los puntos de acceso inalámbrico y algoritmos propietarios para determinar la posición geográfica de un dispositivo móvil con una precisión de entre 20 y 30 metros.

La base de datos de Skyhook se ha construido a través de *wardriving*, e incluye más de 100 millones de puntos de acceso inalámbricos, que llegan a cubrir gran parte de las áreas urbanas tanto de Estados Unidos y Canadá como de numerosos países en Europa, entre ellos España [18]. Además, cualquier usuario puede incluir un nuevo punto de acceso y sus coordenadas a través de la propia Web.

Skyhook también ha desarrollado y patentado un sistema de posicionamiento híbrido, denominado XPS. Este sistema combina las señales WPS, GPS y de las antenas de telefonía para determinar la posición de cualquier tipo de dispositivo móvil. Según sus creadores, XPS es el sistema de posicionamiento más rápido, preciso, fiable y flexible del mercado [19].

El éxito de Skyhook ha sido tal que en Enero de 2008 el presidente de Apple Steve Jobs anunció que tanto iPhone como iPod usarían el sistema WPS de Skyhook como motor de localización para algunas de sus aplicaciones [20]. Igualmente, el 26 de Abril de 2010, Motorola anunció que dejaría de usar los servicios de localización de Google para pasar a usar los de Skyhook [21].

En lo que respecta al desarrollo de aplicaciones, Skyhook ha publicado un SDK con soporte para Linux, Mac OSX, Windows, Symbian y Android [22]. Se

puede acceder al SDK bajo registro gratuito, posibilitando que los desarrolladores puedan habilitar el uso de la localización en sus aplicaciones.

La arquitectura de Skyhook incluye un cliente y un servidor. El cliente actúa como el motor de localización, desde el que una aplicación solicita la localización. Por su parte el servidor provee la información necesaria. Este tipo de arquitectura se ha reproducido de manera similar en el desarrollo de este proyecto.

Resulta interesante comprobar cómo funciona a bajo nivel el API proporcionado por Skyhook. Existen publicados algunos artículos [23] que descubren cómo conocer la posición de cualquier dirección MAC consultando la base de datos de Skyhook, algo que en principio está restringido a las direcciones MAC dentro del rango del dispositivo móvil en cuestión. La comunicación se realiza mediante el envío de datos en formato XML.

2.2.2. Ekahau

Ekahau [24] es un software para la localización en tiempo real de cualquier dispositivo con conectividad Wi-Fi utilizando la infraestructura de redes inalámbricas.

Funciona tanto en interiores como en exteriores, siempre que la cobertura sea adecuada, y puede llegar a localizar un dispositivo con una precisión de hasta 1 metro, en las mejores condiciones. Además, la información de la posición en interiores puede incluir no sólo las coordenadas sino también el piso y la habitación o zona en la que se encuentra. Para ello, es necesario un entrenamiento previo.

Esta fase de entrenamiento — que podría equipararse a las tareas de *wardriving* — es vital para la posterior localización de los dispositivos. Durante esta fase de entrenamiento se analiza la infraestructura de la red y se calibran los mapas que se utilizarán en la fase de localización, midiendo la potencia recibida de los diferentes puntos de acceso en varios puntos.

En esta segunda fase, se utilizan diferentes algoritmos patentados para localizar los dispositivos. Usando la base de datos creada durante el entrenamiento, se comparan las potencias recibidas de los puntos de acceso para crear un mapa de vectores de potencia con el que determinar la ubicación.

Este sistema presenta la desventaja de que, si bien es compatible con cualquier tipo de dispositivo, todos ellos deben llevar instalado el software cliente para que puedan ser localizados.

2.2.3. Google

A finales del año 2007, Google comenzó a ofrecer un nuevo servicio llamado *My Location* — Mi ubicación — [25]. Este servicio, incluido en varios de sus productos como Google Latitude [26] o Google Maps para dispositivos móviles,

tiene la capacidad de conocer en cualquier momento la localización de los usuarios. Una de las características de *My Location* es que no necesita del uso de GPS para conocer la ubicación de un usuario. En su lugar, el sistema utilizó en sus orígenes la información de las torres de telefonía a las que el dispositivo estaba conectado.

En otoño del año 2008, el servicio se actualizó con la inclusión de las redes inalámbricas como un método añadido para determinar la ubicación de un usuario [27], con el objetivo de mejorar su precisión, pues el rango de un punto de acceso es mucho menor al de una torre de telefonía.

Sin embargo, resulta sorprendentemente complicado encontrar información acerca del mismo. Según Google, el servicio de localización mediante redes Wi-Fi mejoraría a medida que más gente lo usase, lo que indica que puede estar usando los dispositivos de los propios usuarios para recolectar nueva información.

Recientemente, Google ha admitido [28] que aprovechó la flota de coches que han recorrido medio mundo tomando fotografías para el servicio Street View, para recoger igualmente datos acerca de las redes inalámbricas descubiertas a su paso, es decir, *wardriving*. Como se describió en la sección 2.1.1, este hecho no supondría mayor problema, de no ser porque también se han recogido muestras de paquetes de datos conteniendo información privada. Esto le ha supuesto a Google incontables problemas legales en todo el mundo [29] [30] [30].

2.2.4. Geolocalización mediante redes Wi-Fi en Android

Todos los dispositivos móviles con sistema operativo Android instalado tienen diferentes maneras de conocer su ubicación en cualquier momento, entre ellas:

- A través de triangulación de señales de telefonía móvil.
- Usando el dispositivo GPS.
- Basándose en las redes inalámbricas a nuestro alrededor.

Estas opciones son utilizadas por muchos de los servicios preinstalados en los dispositivos y que necesitan conocer la localización del mismo. Sin embargo, por razones de privacidad los usuarios deben previamente dar su expreso consentimiento para utilizar estas opciones, pudiendo habilitar y deshabilitarlas siempre que consideren oportuno a través de los ajustes de su dispositivo — ver figura 2.5 — por lo que no todas las opciones están disponibles en cualquier momento.

La opción de geolocalización mediante redes Wi-Fi hace uso del sistema creado por Google con tal fin. De esta manera, cuando la opción de conocer la ubicación del dispositivo está habilitada, y la tarjeta inalámbrica activada, se escanean las redes inalámbricas alrededor del dispositivo para determinar la posición geográfica del mismo. La precisión obtenida puede variar desde decenas hasta cientos de metros.

Figura 2.5: Ajustes de ubicación en un dispositivo Android.

Por último cade destacar que no se ha encontrado en Android Market ningun otro sistema de posicionamiento mediante redes Wi-Fi. Puesto que el propio SDK de Android ofrece acceso a los servicios de localización de Google, cualquier desarrollador puede utilizarlos en sus aplicaciones, de manera que el desarrollo de estos sistemas, una vez existen estas alternativas, no es muy frecuente.

Capítulo 3

Recogida de datos y mapeo de redes Wi-Fi: La aplicación *Warwalk*

Como se mencionó en la sección 2.1.5, la aplicación *Wardrive* desarrollada por Raffaele Ragni se ha usado como base para la aplicación dedicada al descubrimiento, recolección de datos y posicionamiento de redes Wi-Fi. La elección de esta aplicación frente a otras alternativas se basa en diferentes motivos.

En primer lugar, el hecho de que *Wardrive* esté publicado bajo la licencia GNU General Public License [34] permite que cualquier desarrollador pueda conseguir una copia del código fuente de la misma, que puede ser modificada respetando siempre los términos de dicha licencia y publicando el trabajo resultante bajo iguales condiciones. El código fuente completo de la aplicación y el servidor se encuentra publicado en la página del proyecto en Google Code [32] donde está alojado.

Por otro lado, la aplicación cumple con los requisitos principales para el primer paso de este proyecto, la recogida de datos y el uso de un servidor para almacenar los datos, además de ofrecer otro tipo de opciones que otorgan cierta flexibilidad al usuario.

De esta manera, partiendo de la aplicación original se ha creado una nueva aplicación llamada *Warwalk*, adaptada a las necesidades del presente proyecto, que se utilizará para la recolección de datos de redes Wi-Fi, paso previo y necesario para la posterior geolocalización mediante redes Wi-Fi.

En este capítulo se analizará en detalle la aplicación original, describiendo cada una de sus funcionalidades, se debatirá la necesidad de realizar modificaciones en la misma para cumplir los objetivos de este proyecto y se detallarán los cambios acometidos.

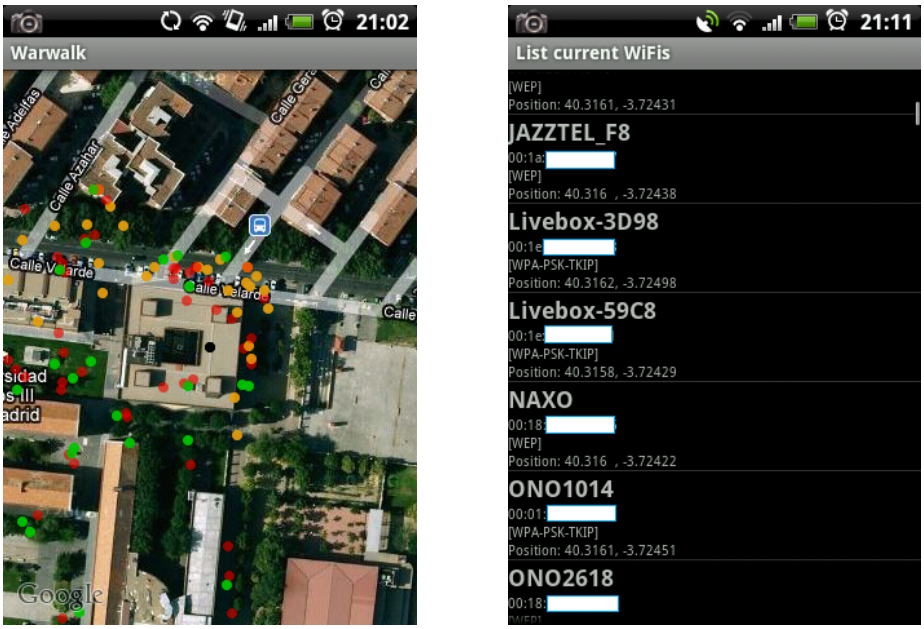

Figura 3.1: Interfaz de la aplicación *Wardrive*

3.1. Análisis de la aplicación original *Wardrive*

3.1.1. Diseño de la interfaz

La interfaz principal de la aplicación resulta muy sencilla, algo habitual en este tipo de aplicaciones. Una vez se lanza la aplicación, se puede comprobar cómo la totalidad de pantalla está ocupada por un mapa en el cual se encuentra marcada la posición del usuario, así como de las redes descubiertas.

La posición del usuario se muestra mediante un punto negro, y habitualmente el mapa se encontrará centrado en ese punto siguiendo el movimiento del usuario, si bien esto puede variar haciendo uso de las opciones de la aplicación.

Las redes Wi-Fi descubiertas hasta el momento se muestran también en el mapa. Aquellas redes que cuenten con algún tipo de encriptación se mostrarán en pantalla con un punto naranja — aquellas con encriptación WEP — o rojo — en cualquier otro caso —, mientras que las redes *abiertas*, es decir, aquellas que no usan ningún tipo de encriptación, lo hacen con un punto verde.

Existe una única vista diferente de la habitual, al margen de los menús de la aplicación. Esta vista se muestra al seleccionar la opción «List current WiFis» en el menú de opciones, y muestra diferente información acerca de las redes Wi-Fi almacenadas en la base de datos. Como se puede comprobar en la figura 3.1, su diseño es también muy sencillo.

3.1.2. Catálogo de opciones

La aplicación ofrece una gran variedad de opciones al usuario, haciendo que muchos de los aspectos sean personalizables. Cuando se despliega el menú de opciones, se pueden encontrar las siguientes posibilidades:

Service ON/OFF — *Servicio ON/OFF* — Pone en funcionamiento o detiene el servicio de detección de redes Wi-Fi.

List current WiFis — *Listar redes Wi-fi actuales* — Muestra en una nueva pantalla la información — BSSID o dirección MAC, SSID, seguridad, frecuencia, nivel de potencia y coordenadas — de las redes Wi-Fi almacenadas que se muestran en ese momento en el mapa de la aplicación, es decir, aquellas dentro del área mostrada.

Statistics — *Estadísticas* — Muestra información acerca del contenido de la base de datos de la aplicación, tal como el número total de redes almacenadas, número de redes abiertas y cerradas, número de redes nuevas detectadas desde la última ejecución y fecha de la última ejecución.

Export to KML — *Exportar a KML* — Exporta toda la información de la base de datos a formato `kml`, creando un fichero que se almacena en la tarjeta de memoria del dispositivo y que puede ser abierto y visualizado con Google Maps o Google Earth.

GPS Times — *Tiempos GPS* — Selección del intervalo mínimo en segundos o metros para el refresco de la señal del GPS. Dicho de otro modo, el tiempo transcurrido o la distancia recorrida mínima antes de solicitar una nueva localización al dispositivo GPS. Existen tres opciones, intervalo corto — 3 segundos o 3 metros —, medio — 10 segundos o 10 metros — y largo — 30 segundos o 50 metros —.

Send to online DB — *Enviar a base de datos online* — Envía los datos almacenados hasta el momento a la base de datos online de Wardrive [33]. Seleccionando esta opción se muestra en pantalla un diálogo desde el que se puede decidir si enviar todas las redes o únicamente aquellas nuevas respecto de la última vez que se enviaron datos.

WiFi Filter — *Filtro Wi-Fi* — Permite la creación de filtros para las redes Wi-Fi, de manera que se ignoren determinadas redes en función de los parámetros propuestos.

Map — *Mapa* — Habilita o deshabilita el modo satélite de Google Maps, es decir, la descarga de imágenes satélite. Si está deshabilitado se muestra únicamente el mapa de la zona.

Labels — *Etiquetas* — Determina si se muestra en el mapa una etiqueta con el SSID de cada red para ayudar a su identificación.

Follow — *Seguimiento* — Determina si el mapa sigue el movimiento del usuario automáticamente o bien es este quien maneja el movimiento del mismo.

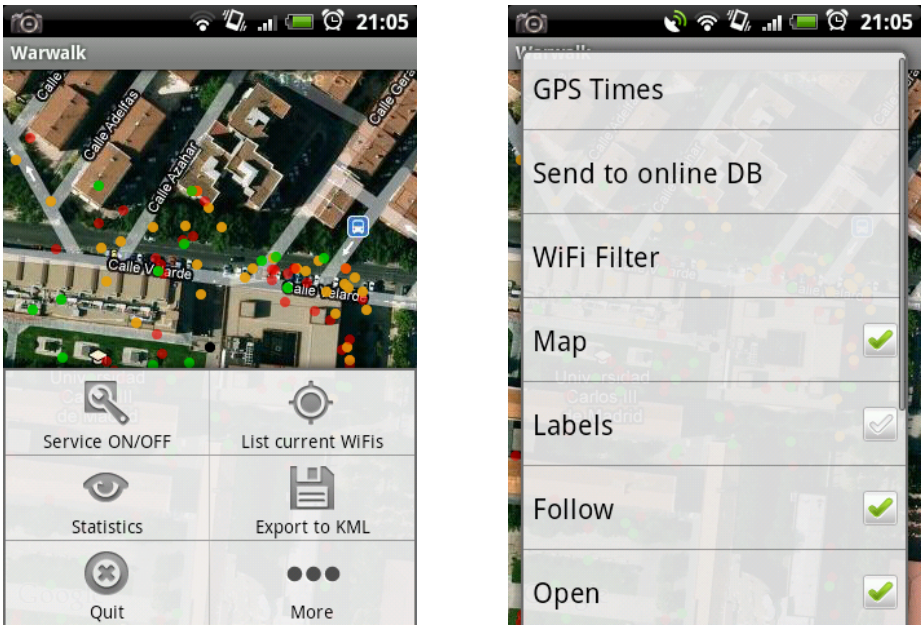

Figura 3.2: Menú de opciones de la aplicación Wardrive

Open — *(Redes) Abiertas* — Muestra u oculta en el mapa los puntos correspondientes a las redes abiertas, es decir aquellas sin ningún tipo de encriptación.

Closed — *(Redes) Cerradas* — Muestra u oculta en el mapa los puntos correspondientes a las redes cerradas, esto es, aquellas que están encriptadas.

Notifications Enabled — Notificaciones habilitadas — Habilita o deshabilita las notificaciones en pantalla.

About — *Acerca de* — Muestra información acerca de la aplicación, su desarrollador y la licencia de uso.

Delete ALL — *Borrar todo* — Elimina la totalidad de redes almacenadas en la base de datos hasta ese momento. Se muestra al usuario un diálogo de confirmación previo a realizar la operación.

Quit — *Terminar* — Detiene el servicio y finaliza la ejecución del programa.

En la figura 3.2 se puede comprobar el aspecto del menú de opciones. Cuando el usuario despliega el menú se muestra la captura de la izquierda, mientras que la segunda captura, a la derecha, se muestra cuando se selecciona la opción «*More...*» del menú. Para más detalles acerca de la construcción de menús en Android, se recomienda consultar la sección 4.2.2.

3.1.3. Algoritmo de posicionamiento

Al igual que otras aplicaciones similares, la aproximación que toma Wardrive a la hora de posicionar los puntos de acceso es la más sencilla posible. En realidad, las posiciones almacenadas son aquellas desde las que se han observado esos puntos de acceso. Dado que es muy probable que una red se encuentre dentro del alcance del dispositivo en diferentes ocasiones durante el movimiento del usuario, existen varios puntos desde los que la red ha sido observada. Wardrive utiliza entonces el nivel de potencia recibida de la señal para elegir aquel punto desde el que se recibió la señal con mayor potencia, siendo éstas las coordenadas que se almacenan en la base de datos.

Esta aproximación puede funcionar bien para una actividad como el *wardriving*, donde la precisión no es un aspecto determinante y los datos se recogen desde un vehículo en movimiento, con una mayor distancia entre puntos y con una ruta delimitada — las propias calles —.

Sin embargo también presenta diferentes inconvenientes, como la imposibilidad de encontrar puntos de acceso con coordenadas diferentes a aquellas por las que el usuario haya efectivamente transitado durante su recorrido, lo que a su vez dificulta que el mapa de datos de acceso se distribuya de manera uniforme, ofreciendo un aspecto lineal y alejado de la realidad.

3.1.4. Mapeo de redes

Como suele ser habitual en aplicaciones de este tipo, se incluye una herramienta para mapear los resultados. En el caso de *Wardrive*, la base de datos puede ser exportada a KML, formato compatible con diferentes herramientas de Google como Google Maps o Google Earth, si bien otras aplicaciones pueden igualmente soportarlo.

El formato de un fichero KML es muy similar al de cualquier fichero XML, lenguaje en el que está basado. Un fichero KML especifica un conjunto de datos con unas coordenadas asociadas que se mostrarán en el mapa.

La muestra de código 1 incluye un ejemplo del formato de los archivos KML.

3.1.5. Base de datos online

Wardrive también ofrece a sus usuarios la posibilidad de subir los datos recogidos al servidor propio. Estos datos están disponibles para su consulta en la página web de la aplicación [33], donde se muestra la localización de todos los puntos de acceso subidos hasta el momento usando Google Maps, tal y como se muestra en la figura 3.3.

El servidor aloja un servlet que es el encargado de gestionar toda la información recibida, y cuyo código también se encuentra disponible en la página del proyecto de la aplicación en Google Code.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
    <Placemark>
        <name><![CDATA[WiFi-UC3M]]></name>
        <description><![CDATA[...]]></description>
        <Point>
            <coordinates>-3.7253,40.3157,686.0</coordinates>
        </Point>
    </Placemark>
    <Placemark>
        <name><![CDATA[WebSTAR]]></name>
        <description><![CDATA[...]]></description><styleUrl>
        <Point>
            <coordinates>-3.5222,40.4156,661.0</coordinates>
        </Point>
    </Placemark>
</Document>
</kml>
```

Listing 1: Muestra del formato de un fichero KML

Además, en las versiones más recientes de Wardrive — en conreto a partir de la versión 3.2 — se incluye también la posibilidad de subir los datos a los servidores de WiGLE, si bien la versión en la que se ha basado este proyecto es anterior a la misma y por tanto no incluye esta opción.

3.1.6. GNU - General Public License

Como se citó al comienzo de este capítulo, la aplicación *Wardrive* está distribuida bajo la licencia GNU General Public License — de ahora en adelante, GNU GPL —, de manera que se trata de software libre. La licencia GNU GPL tiene como objetivo garantizar la libertad del software para compartir y modificar todas las versiones del mismo y asegurar que permanecerá como software libre para todos sus usuarios. De este modo, cuando se modifica un trabajo, se asume la responsabilidad de respetar las libertades originales del mismo.

Se ha sido especialmente cuidadoso a la hora de respetar los términos de la licencia, incluyendo los avisos legales necesarios para indicar que se trata de un trabajo modificado basado en la fuente original.

Es posible consultar en línea el texto completo original de la licencia GNU GPL en inglés [34] así como una traducción no oficial al español [35].

Figura 3.3: Página web de la base de datos *Wardrive*

3.2. Versión modificada: *Warwalk*

Tras estudiar a fondo la versión disponible de *Wardrive* y sus características, se ha llegado a la conclusión de que, si bien presenta la funcionalidad requerida para el proyecto, también es necesario mejorar y/o adaptar ciertos aspectos para ajustarlos a las necesidades concretas del mismo.

La base de la aplicación permanece inalterada, y aspectos como el catálogo de opciones o la interfaz gráfica no se han visto modificados. Principalmente son tres los aspectos sobre los que se ha realizado algún tipo de modificación, y que han supuesto cambios en diversos componentes de la aplicación:

- La implementación de un nuevo algoritmo para el posicionamiento de puntos de acceso inalámbricos.
- El rediseño de la base de datos de la aplicación.
- La implementación de un servidor donde se almacenarán los datos.

En primer lugar, se ha implementado un nuevo algoritmo para el posicionamiento de los puntos de acceso. Como se ha descrito en el apartado anterior, la aplicación se limitaba a almacenar las coordenadas GPS desde donde se ha recibido la señal de cierta red, escogiendo siempre como posición definitiva aquel punto desde el cual la señal se recibiese con mayor intensidad. En su lugar, se ha implementado un nuevo algoritmo de triangulación o medias ponderadas que pretende localizar con mayor precisión el foco de la señal inalámbrica.

Por otra parte, el nuevo algoritmo ha traido consigo la necesidad de rediseñar parte de la base de datos para adaptarse a las nuevas necesidades. En concreto, diversos campos se han añadido a la tabla con nueva información acerca de las redes inalámbricas.

Por último, uno de los objetivos de este proyecto es el de implementar un servidor donde almacenar los datos en base a los cuales ser capaz de determinar la posición geográfica de un usuario. Pese a que Wardrive cuenta con su propio servidor, su uso no es viable para el proyecto, por lo que ha resultado necesario implementar uno propio adapatado a los requisitos propuestos.

A continuación se describirán en detalle cada uno de los cambios en los diferentes componentes de la aplicación, justificando cada una de las decisiones tomadas durante el proceso.

3.2.1. Algoritmo de posicionamiento

Como ya se decribió en la sección 3.1.3, la manera en que Wardrive aproxima la posición de los puntos de acceso descubiertos es muy sencilla, limitándose a grabar las coordenadas GPS del punto desde el que se ha recibido el mayor nivel de señal.

Por tanto, una de las prioridades a la hora de diseñar e implementar el nuevo algortimo no era otra que mejorar tanto la precisión en el posicionamiento como la distribución de los puntos de acceso en el mapa, y en definitiva buscar un diseño mejor adaptado a la recogida de datos a pie, sin renunciar a otros métodos.

La solución para este problema, tras valorar diferentes alternativas, ha pasado por implementar un sistema de triangulación basado en medias, muy similar al que utilizan otras herramientas presentadas con anterioridad, como WiGLE o Kismet. Fundamentalmente, dichos algoritmos asumen que el nivel de potencia de la señal inalámbrica será inversamente proporcional a la distancia del usuario al foco de la misma. Por tanto, se pondera cada medición en función de la potencia recibida en ese punto geográfico.

De esta manera, el cálculo de las coordenadas geográficas — latitud y longitud — para cada red Wi-Fi detectada viene determinado por las siguientes fórmulas:

$$latitud = \frac{\sum_{i=1}^{n} latitud_i * 1/nivel_i}{\sum_{i=1}^{n} 1/nivel_i}$$

Y de manera análoga para el cálculo de la longitud:

$$longitud = \frac{\sum_{i=1}^{n} longitud_i * 1/nivel_i}{\sum_{i=1}^{n} 1/nivel_i}$$

donde:

- *latitud* y *longitud* son las coordenadas geográficas finales asignadas al punto de acceso.
- n es el número total de mediciones realizadas para una única red.
- $latitud_i$ y $longitud_i$ son las coordenadas recogidas en una única medición y para una única red.
- $nivel_i$ es el nivel de potencia de la señal de una red recogido en una medición determinada.

Como se puede comprobar, se utiliza el inverso del nivel de potencia de señal. La explicación a este hecho es que el valor de la intensidad de potencia ofrecido por Android se mide en dBm.

Un dBm se define como el nivel de potencia en decibelios en relación a un nivel de referencia de un milivatio (1mW). Este valor aumenta cuanto mayor es la potencia. Además, cuando se usa para medir la potencia de redes inalámbricas su valor es negativo, ya que la potencia recibida por la antena inalámbrica es habitualmente inferior a 1mW. Como referencia, se puede considerar que la potencia máxima recibida de una red inalámbrica varía típicamente entre los

-10dBm y los -30 dBm, mientras que los valores habituales se sitúan entre -60dBm y -80dBm [36].

Resulta por tanto necesario utilizar el inverso de este valor como ponderación, ya que de otra manera se lograría precisamente el efecto contrario, otorgando un mayor peso a las coordenadas desde las que el nivel de potencia de la señal era menor.

Este algoritmo también presenta ciertos inconvenientes. Debido al sistema de promedio de las coordenadas, los resultados no son tan precisos cuando la señal se recibe siempre desde el mismo lado, pues tienden a desplazarse en demasía. Los mejores resultados aparecen cuando es posible rodear los puntos de acceso, caminando alrededor de los edificios, pues es en estos casos cuando el algoritmo funciona mejor y se saca el mayor provecho al promedio de las coordenadas.

3.2.2. Base de Datos

Dadas las características del nuevo algoritmo propuesto para el posicionamiento de los puntos de acceso descrito en el apartado anterior, la base de datos SQLite donde se almacena la información recogida durante la ejecución de la aplicación tuvo que ser modificada para dar cabida a nuevos campos.

Dado que el cálculo de las coordenadas para una red inalámbrica supone realizar varios sumatorios, se necesita información relativa a cada una de las mediciones efectuadas para todas las redes detectadas. Por tanto se ha de plantear una manera de contar con esa información disponible en todo momento. Las alternativas barajadas se reducen a las siguientes:

- Almacenar en una nueva tabla de la base datos la información relativa a las coordenadas — latitud y longitud — y el nivel de potencia de la señal por cada medición realizada y red detectada.
- Realizar el cálculo y la actualización de los sumatorios *al vuelo* para cada red detectada, esto es, en el momento de recoger una nueva medición.

La primera opción tiene el inconveniente de la ingente cantidad de datos que habría que almacenar en la base de datos. Sería necesario crear una nueva tabla donde almacenar las coordenadas recogidas en todas las mediciones realizadas. Además también supondría un aumento en el coste computacional, al tener que realizarse el cálculo de los sumatorios para una red en todas las ocasiones en las que se recogiese una nueva medición de la misma.

La segunda alternativa se ha considerado más viable. El valor de los diferentes sumatorios necesarios para calcular las coordenadas se almacenan en la base de datos, por lo que únicamente es necesario actualizarlos cada vez que se recoge una nueva medición. El coste computacional, por tanto, es menor, y la base de datos no aumenta de tamaño excesivamente, ya que simplemente se necesitan añadir nuevas columnas a la única tabla ya existente.

Por tanto, se decidió implementar esta última opción, realizando las modificaciones necesarias en la base de datos. Se han añadido a la tabla con la información de las redes estos tres campos:

- El campo *sumwlat*, de tipo *real*, que almacena el valor del sumatorio de la latitud, ponderada por el inverso del nivel de potencia de la señal.
- El campo *sumwlon*, que análogamente almacena el valor del sumatorio de la longitud, ponderada por el inverso del nivel de potencia de la señal.
- El campo *sumlevel*, de nuevo de tipo *real*, donde se guarda el valor del sumatorio del inverso de los niveles de potencia obtenidos.

De esta manera, cada medición realizada de una red supone actualizar cada uno de estos tres valores en la entrada correspondiente de la base de datos, además de los campos de latitud y longitud una vez calculados sus nuevos valores en base al nuevo algoritmo propuesto en el apartado anterior.

Tras estos cambios, la base de datos utilzada por la aplicación consta de una única tabla denominada *networks*, y que contiene los siguientes campos:

bssid La dirección MAC del punto de acceso.

ssid La SSID de la red.

capabilities Las características de la red, en concreto el tipo de encriptación utilizada.

level El nivel de potencia de la señal máximo detectado de la red.

frequency La frecuencia en la que emite la red.

lat La latitud de la red, en grados.

lon La longitud de la red, en grados.

alt La altitud sobre el nivel del mar de la red, en metros.

timestamp El momento en el que la red fue detectada por última vez.

sumwlat El sumatorio de la longitud, ponderada por el inverso del nivel de potencia de la señal.

sumwlon El sumatorio de la latitud, ponderada por el inverso del nivel de potencia de la señal.

sumlevel El sumatorio del inverso de los niveles de potencia detectados de la red.

El script para la creación de la base de datos de la aplicación se presenta en la muestra de código 2.

```
CREATE TABLE IF NOT EXISTS networks (
    bssid text primary key,
    ssid text,
    capabilities text,
    level integer,
    frequency integer,
    lat real,
    lon real,
    alt real,
    timestamp integer,
    sumwlat real,
    sumwlon real,
    sumlevel real
)
```

Listing 2: Creación de la base de datos de la aplicación

3.2.3. Servidor

El último cambio a acometer sobre la aplicación original *Wardrive* ha sido la implementación de un nuevo servidor. Este servidor tendrá dos funciones fundamentales, por un lado almacenará los datos recogidos por la aplicación *Warwalk* que sean enviados por los usuarios, de la manera que será detallada a continuación, y por otra parte enviará dicha información a los usuarios para determinar su posición geográfica, función que se describirá en el siguiente capítulo.

En lo que se refiere a la aplicación que nos ocupa, *Warwalk*, la función del servidor es la de permanecer a la escucha de conexiones entrantes por parte de aquellos usuarios que deseen enviar los datos que hayan recogido con la aplicación, y almacenar dichos datos en una base de datos para su posterior consulta.

Tres son los componentes básicos del servidor, que se describirán detalladamente en las siguientes secciones:

- Una base de datos que almacena toda la información enviada por los propios usuarios.
- Una serie de scripts PHP que procesan la información recibida del cliente y actúan en consecuencia.
- El servidor web propiamente dicho, donde están alojados los scripts y sobre el cual son ejecutados.

Base de Datos

En cuanto a la base de datos, se ha optado por MySQL, versión 5.1.36. Sobre él se ha implementado una base de datos llamada *wps*, que cuenta con

tres tablas, de nombres *networks*, *updates* y *query*.

La tabla *networks* es la de mayor importancia y en donde se almacena la información que *a posteriori* será consultada para determinar la posición geográfica de un usuario. En ella están almacenados, entre otros, el *bssid* o dirección MAC que identifica al punto de acceso y sus coordenadas.

```
CREATE TABLE IF NOT EXISTS `networks`(
    'bssid' varchar(17) NOT NULL,
    'ssid' varchar(128) NOT NULL,
    'capabilities' text NOT NULL,
    'lat' double NOT NULL,
    'lon' double NOT NULL,
    'alt' double NOT NULL,
    'confidence' int(2) NOT NULL DEFAULT '99',
    PRIMARY KEY (`bssid`)
)
```

Listing 3: Creación de la tabla *networks* de la base de datos *wps*

Se puede comprobar cómo existe un campo de la tabla que no guarda relación con la base de datos de la aplicación de la que proceden los datos. Se trata del campo *confidence*, cuyo propósito y funcionamiento se describirán en el próximo apartado.

Por su parte, la tabla *updates* almacena todas y cada una de las actualizaciones realizadas por los usuarios de la aplicación. Esta información no es utilizada por ninguna de las aplicaciones sino que se guarda principalmente con fines estadísticos y de control. Una vez procesada la información, sería posible determinar qué redes son actualizadas con mayor frecuencia o en qué zonas geográficas existen más usuarios de la aplicación, entre otras aplicaciones.

```
CREATE TABLE IF NOT EXISTS 'updates' (
    'id' int(11) NOT NULL AUTO_INCREMENT,
    'bssid' text NOT NULL,
    'lat' double NOT NULL,
    'lon' double NOT NULL,
    'timestamp' double NOT NULL,
    PRIMARY KEY ('id')
)
```

Listing 4: Creación de la tabla *updates* de la base de datos *wps*

Por último, la tabla *query* almacena las posibles operaciones a realizar en la base de datos, desde una consulta a operaciones de actualización o inserción. Los datos de esta tabla son utilizados por una de las librerías usadas en la implementación de los *script* PHP, desarrolada por Moisés Martínez Muñoz.

```
CREATE TABLE IF NOT EXISTS 'query' (
    'ID' int(4) NOT NULL AUTO_INCREMENT,
    'Nombre' varchar(75) NOT NULL,
    'Etiqueta' varchar(75) NOT NULL,
    'EtiquetaNodo' varchar(75) NOT NULL,
    'Consulta' text NOT NULL,
    'Tipo' int(4) NOT NULL DEFAULT '0',
    'Nodos' int(4) NOT NULL DEFAULT '0',
    PRIMARY KEY ('ID')
)
```

Listing 5: Creación de la tabla *query* de la base de datos *wps*

Scripts PHP

Se ha decidido utilizar PHP versión 5.3.0 como lenguaje para la creación de los distintos *scripts* alojados en el servidor. Cuando el servidor recibe una petición de un usuario que desea subir los datos recogidos con la aplicación, uno de estos *scripts* recibe los datos de una única red para almacenarlos en la base de datos, siguiendo una cierta lógica.

El primer paso es determinar si la red es nueva o ya se encuentra en la base de datos, para lo que es suficiente una sencilla consulta a la misma. En cualquiera de los dos casos, la actualización se almacena y queda registrada en la tabla *updates* de la base de datos.

En el caso de que la red no se encuentre en la base de datos, los datos se insertarán como una nueva fila en la tabla *networks*, sin mayores consecuencias.

Sin embargo, cuando la red ya se encuentra almacenada, es necesario llegar a un compromiso entre los datos ya almacenados y los que se acaban de recibir, que pueden ser diferentes. Es en este punto cuando el campo *confidence* de la tabla *networks*, mencionado en el apartado anterior, entra en juego y resulta especialmente interesante recalcar la manera en que funciona.

Determinar la posición geográfica exacta de un determinado punto de acceso supone un reto. Si bien se ha intentado conseguir la mayor precisión posible a través del nuevo algoritmo de la aplicación, aún existen otras cuestiones que la aplicación no puede manejar pero sí se hacen patentes una vez la información se envía al servidor.

Diferentes usuarios usando la aplicación *Warwalk* pueden obtener coordenadas ligeramente diferentes para un determinado punto de acceso, en función del recorrido que hayan realizado o el número de mediciones realizadas. Incluso las condiciones atmosféricas pueden alterar los niveles de potencia recibidos de una señal y, por tanto, los resultados de la aplicación. Por otra parte, la posición geográfica de un punto de acceso puede no ser fija y variar con el tiempo, ya que su ubicación física puede modificarse. Si se da el caso, los usuarios pueden

reportar una ubicación completamente distinta de la almacenada para un punto de acceso, mientras que la posición almacenada en el servidor pasaría a ser incorrecta.

Para lidiar con estas cuestiones se ha implementado un sistema de confianza en las coordenadas de todas las redes inalámbricas, cuyo objetivo es determinar el grado de veracidad que tiene en un momento determinado las coordenadas de una red almacenadas en el servidor, y cuyo valor se almacena como ya se ha comentado en la columna *confidence* de la tabla *networks*.

De este modo, cada vez que un usuario envía datos de una red que ya se encuentra en la base de datos, se comprueba si las coordenadas recibidas se encuentran dentro de un área cercana al punto ya almacenado. Si la respuesta es positiva, la confianza en la red aumenta, o dicho en términos de la implementación, el valor del campo *confidence* aumenta en una unidad. En caso contrario, la confianza decrece y el valor de *confidence* se decrementa. El valor mínimo de confianza en una red es de 0 y el máximo es de 99, de manera que se limita el efecto que una actualización puede provocar en los valores almacenados de las coordenadas de una determinada red.

La manera en que las coordenadas recibidas afectan a las presentes en la base de datos es siempre la misma, y depende precisamente de la confianza en la red en cuestión. La fórmula para calcular el valor de la longitud de una red es la siguiente:

$$latitud = \frac{latitud * confianza + latitud_{nueva} * (100 - confianza)}{100}$$

Y de igual manera, para el cálculo de la longitud:

$$longitud = \frac{longitud * confianza + longitud_{nueva} * (100 - confianza)}{100}$$

donde:

- *latitud* y *longitud* son los valores de las coordenadas almacenados en la base de datos.
- $latitud_{nueva}$ y $longitud_{nueva}$ son los valores de las coordenadas reportados por la nueva actualización.
- *confianza* es el valor de confianza en las coordenadas de la red almacenado en la base de datos.

Con este sistema se ha conseguido un gran equilibrio en la manera en la que se gestiona la posición de las redes. Mientras las actualizaciones para una red reporten localizaciones cercanas, la confianza en la misma aumentará hasta su máximo. Una única actualización con una posición alejada a la almacenada, y que por tanto podría considerarse errónea, apenas modificará el valor ya almacenado. Sin embargo, si se suceden las actualizaciones con valores alejados pero

similares entre sí — lo que podría indicar un cambio en la ubicación del punto de acceso —, la confianza en la red decrecerá paulatinamente, y las sucesivas actualizaciones tendrán un mayor efecto en el cálculo de las coordenadas. Dado que ambos valores — el almacenado y el de la actualización — tenderán a convenger, finalmente la confianza volverá a aumentar y el ciclo puede volver a repetirse.

Servidor web

Por último, el servidor web utilizado es Apache Tomcat en su versión 2.2.11, de distribución gratuita. El servidor permanece siempre a la espera de recibir conexiones HTTP, y sobre él son ejecutados los scripts PHP. Las comunicaciones se realizan a través del protocolo HTTP. Además, las peticiones HTTP utilizan el método POST para el envío de datos, sobre el que se encapsula la información.

Capítulo 4

Geolocalización mediante redes Wi-Fi: La aplicación WPS

En este capítulo se describirá en detalle el desarrollo completo de la aplicación *WPS* para el posicionamiento del usuario basado en los datos de redes Wi-Fi recogidos y almacenados previamente en el servidor mediante la aplicación *Warwalk*.

El nombre de *WPS* viene de *Wireless Positioning System*, las siglas en inglés para definir un sistema de posicionamiento mediante redes inalámbricas.

A lo largo de este capítulo se harán numerosas referencias a muestras del código original de la aplicación. Este código no tiene por qué ser ejecutable y ha sido modificado y simplificado con el objetivo de ofrecer al lector una visión general de la manera en que se implementan las diferentes funcionalidades.

4.1. Análisis y diseño

4.1.1. Introducción a WPS

Como se ha comentado previamente, el objetivo principal de *WPS* es el de localizar al usuario mediante redes Wi-Fi, utilizando los datos recogidos previamente a través del uso de la aplicación *Warwalk* y que han sido subidos y almacenados en un servidor. Para ello, se hace uso de los siguientes servicios ofrecidos por Android:

- Google Maps, para mostrar en un mapa la ubicación del usuario.
- El dispositivo Wi-Fi, para escanear en busca de señales Wi-Fi y poder determinar la ubicación del usuario.

- Conexión a internet, para poder intercambiar datos con el servidor.
- El dispositivo GPS, para determinar el error en la posición del usuario.

A través de Google Maps es posible mostrar mapas en la aplicación, de manera que el usuario pueda conocer visualmente su posición geográfica y no limitarse a una serie de coordenadas. Este servicio está disponible a través de un paquete específico. Se detallará el funcionamiento de Google Maps en la sección 4.2.3 y siguientes.

El dispositivo Wi-Fi es también una parte indispensable en la aplicación, dado el propósito de determinar la posición del usuario únicamente basados en las redes Wi-Fi que el dispositivo encuentra a su alrededor. Se cubrirán los aspectos relacionados con el manejo del dispositivo Wi-Fi en la sección 4.2.5.

La conexión a internet es necesaria con el fin de poder consultar datos en el servidor, dado que en él se encuentran almacenadas las posiciones de todas las redes Wi-Fi conocidas hasta el momento. El intercambio de información entre la aplicación y el servidor se realiza bajo el protocolo HTTP y mediante el estándar XML. Las conexiones HTTP con el servidor se detallarán en la sección 4.2.8.

El uso de GPS es opcional y se ha incluido en esta versión de la aplicación principalmente por motivos de evaluación de la misma, de manera que se puedan obtener medidas del error cometido por la aplicación a la hora de determinar la ubicación del usuario. Dado que una de las ventajas de la aplicación radica precisamente en ser una alternativa al propio GPS, su uso en otras circunstancias haría que la aplicación careciese de sentido, si bien el uso de GPS no es posible bajo determinadas condiciones, como por ejemplo en interiores de edficios. Se hablará acerca de la gestión del dispositivo GPS en Android en la sección 4.2.6.

En la actualidad la práctica totalidad de dispositivos Android — exceptuando alguno de los primeros terminales puestos a la venta que ya se encuentran prácticamente descatalogados — incorpora entre otros todos estos servicios, por lo que la aplicación puede no sólo ejecutarse sino funcionar correctamente en cualquiera de ellos.

4.1.2. Casos de uso

En este apartado se incluyen dos diagramas de casos de uso utilizando el estándar UML versión 2.3 [37]. el objetivo de estos diagramas es el de ayudar al lector a comprender la funcionalidad típica que se espera que la aplicación desempeñe, describiendo las interacciones entre los actores y el sistema.

El primer diagrama, presentado en la figura 4.1, define la interacción entre el usuario del sistema, que actúa como actor, y el propio sistema — es decir, la aplicación WPS —. A continuación se ofrece una breve descripcion de cada uno de los casos de uso, representados en el diagrama por un óvalo.

- Modificar periodo de escaneo: El usuario será capaz de controlar la frecuencia, o dicho de otro modo el periodo en segundos con el que desea que se actualice su localización.

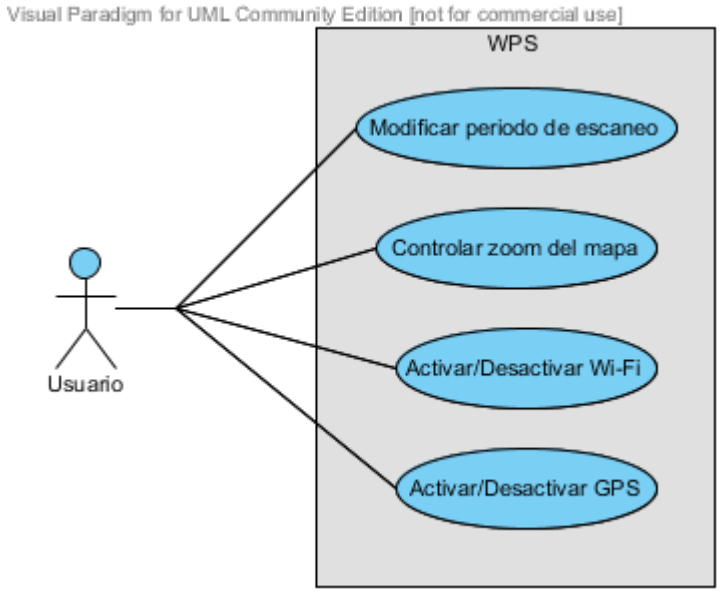

Figura 4.1: Diagrama de casos de uso de WPS

- Controlar zoom del mapa: El usuario debe tener la posibilidad de controlar el nivel de zoom del mapa donde se muestra su ubicación, siendo capaz de aumentar y disminuir el nivel de zoom en todo momento.
- Activar/Desactivar Wi-Fi: El usuario tendrá la opción de activar y desactivar el dispositivo Wi-Fi en cualquier momento, si bien desactivarlo supone la pérdida de las actualizaciones en la localización del mismo.
- Activar/Desactivar GPS: El usuario podrá activar y desactivar el dispositivo GPS en todo momento, con el fin de obtener un valor preciso del error cometido por la aplicación.

En el caso de la figura 4.2, que contiene el segundo de los diagramas, el actor representado es la propia aplicación WPS, quien realiza un único caso de uso al interactuar con el servidor de la aplicación:

- Actualizar localización: La aplicación deberá conectar con el servidor para consultar y descargar los datos necesarios para actualizar la localización del usuario.

4.1.3. Diagrama de clases

Con el objetivo de ayudar al lector a comprender la futura implementación de la aplicación *WPS*, la figura 4.3 muestra el diagrama de clases de la aplicación

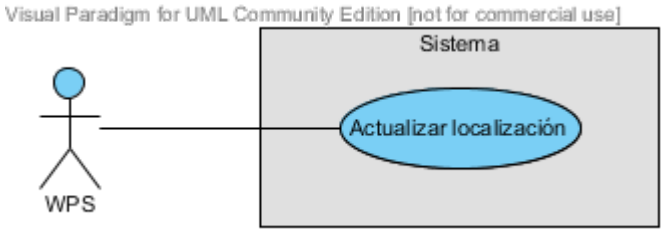

Figura 4.2: Diagrama de casos de uso del servidor de WPS

WPS. Como en el caso de los diagramas de casos de uso, se ha seguido el estándar UML versión 2.3. Se ha optado por no incluir atributos ni métodos en el diagrama de manera que el diagrama resulte lo más sencillo posible, pues como se ha comentado previamente, el único objetivo es el de dar al lector una idea general de cómo se ha abordado el problema.

A continuación se describen brevemente las clases que forman parte de la aplicación:

`WPSApp` Es la clase principal del sistema y la que coordina al resto. Extiende la clase `MapActivity` y muestra al usuario un mapa con su localización y diversa información en pantalla, además de controlar el resto de funcionalidades disponibles para el usuario.

`Constants` Define varias constantes que se usan en el resto de clases.

`ScanResultSortByLevel` Clase que implementa la interfaz `Comparator` y que se usa para definir los criterios a la hora de ordenar los resultados de un escaneo en busca de puntos de acceso inalámbricos.

`XMLHandler` Procesa y extrae la información necesaria de los documentos XML obtenidos como respuesta del servidor de la aplicación.

`RequestPositionOnline` Se trata de la clase que gestiona todo el proceso de peticiones al servidor y actualizaciones de la localización del usuario. Extiende la clase `Thread`, lo que indica que se trata de un proceso independiente.

`CustomLocationOverlay` Extiende a la clase `Overlay` y es la clase encargada de dibujar y actualizar los elementos en el mapa mostrado en pantalla.

4.1.4. Arquitectura

Una vez se tiene una idea general del funcionamiento y características generales de la aplicación, la figura 4.4 ofrece un diagrama que contiene la arquitectura de todo el sistema completo.

Como se puede observar, el sistema completo consta de los diversos elementos, cuya descripción y propósito se detallan a continuación:

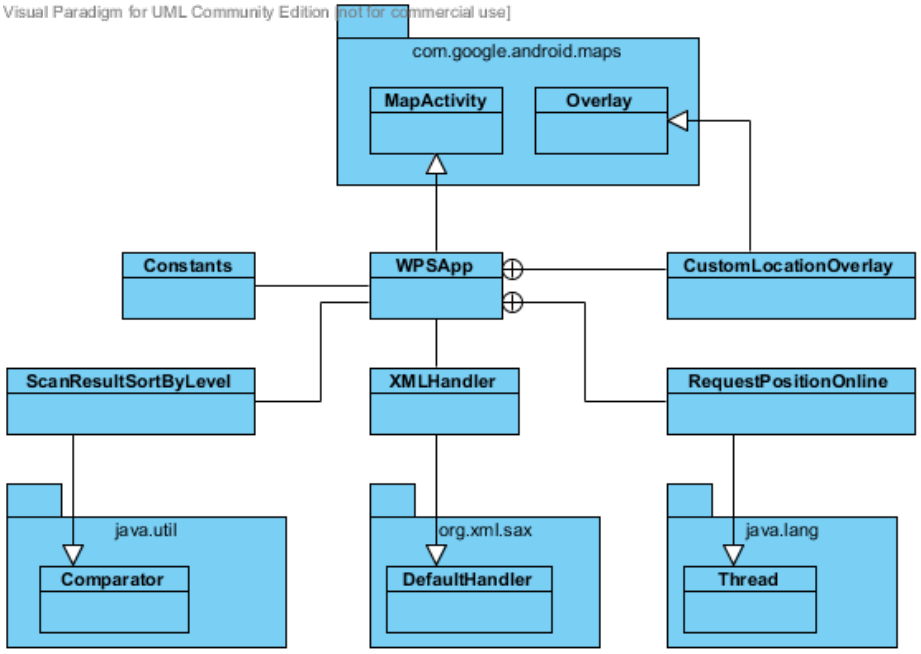

Figura 4.3: Diagrama de clases de la aplicación WPS

- Dispositivo móvil: un dispositivo con sistema operativo Android y con capacidad Wi-Fi y opcionalmente GPS. El uso de Wi-Fi es imprescindible para ubicar al usuario.
- Documentos XML: se utilizan para intercambiar información entre el dispositivo móvil y el servidor.
- Servidor: servidor web donde se aloja la base de datos. Permanece continuamente a la espera de conexiones por parte de cualquier dispositivo.
- Documentos PHP: recibe las peticiones de conexión de un dispositivo y actúa como interfaz para el acceso a la base de datos.
- Base de datos: Almacena los datos de todas las redes Wi-Fi conocidas hasta el momento.

4.1.5. La plataforma Android

Para ayudar al lector en la comprensión de las peculiaridades de la plataforma Android y de los recursos que se han utilizado a la hora de desarrollar la aplicación, en esta sección se tratarán brevemente — ya que el desarrollo para Android no es el objetivo principal de este proyecto — algunos de los aspectos básicos que caracterizan a este sistema operativo.

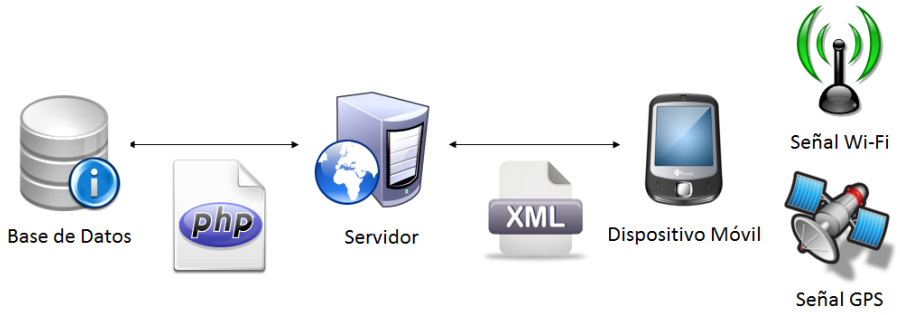

Figura 4.4: Arquitectura de la aplicación

Android [38] es una plataforma software y un sistema operativo creado por Google y la Open Handset Alliance [39], un grupo de empresas de tecnología cuyo objetivo es la innovación en tecnología móvil. Una de las principales características de Android es que se trata de un sistema *open source*, con lo que cualquiera puede descargarse el código fuente y modificarlo a su discrección.

Para el desarrollo de este proyecto, se ha utilizado la versión 2.1 del SDK de Android, no pudiéndose garantizar que la información presentada en este documento sea correcta para cualquier otra versión.

Arquitectura

En resumen, Android es una pila de software ejecutado sobre la versión 2.6 del kernel o núcleo de Linux, cuyas aplicaciones se ejecutan sobre una máquina virtual optimizada para dispositivos móviles, denominada *Dalvik Virtual Machine*. En la figura 4.5 se puede observar gráficamente cómo se estructuran las cuatro diferentes capas de software, que se describen con más detalle a continuación:

Aplicaciones Se trata del más alto nivel, donde se representan todas las aplicaciones, tanto las incluidas en Android como aquellas desarrolladas por otros usuarios. Todas las aplicaciones para Android estan escritas en el lenguaje de programación Java [40].

Marco de aplicaciones En este nivel están representadas las API que dan acceso a las funciones de bajo nivel. Todas las aplicaciones desarrolladas para Android tienen acceso a las mismas API, que también están escritas en Java.

Librerías Esta capa está compuesta por un gran número de librerías C/C++ que ofrecen la mayor parte de las funcionalidades de Android, entre ellas la representación de gráficos, el uso de SQLite, el soporte multimedia, etc.

Entorno de ejecución Se sitúa al mismo nivel que las librerías, e incluye tanto librerías de Java como la máquina virtual *Dalvik*.

Figura 4.5: Arquitectura de Android

Núcleo de Linux La capa de más bajo nivel corresponde al núcleo de Android. Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes.

Acerca de la máquina virtual *Dalvik* cabe destacar que es una máquina virtual diseñada específicamente para sacar el mayor partido al diseño de los dispositivos móviles, en los que la capacidad de procesamiento y memoria es limitada. En lugar de utilizar directamente el bytecode de Java, la máquina virtual *Davik* combina los archivos `.class` en un archivo ejecutable *Dalvik* `.dex`, que puede contener diferentes clases. Después, este archivo se comprime en un paquete para Android `.apk`, que será el fichero distribuido. Además, la máquina virtual utiliza registros y no una pila para almacenar datos. De esta manera se optimiza el espacio consumido y el acceso a los recursos.

Componentes

Los cuatro componentes principales alrededor de los que se implementa cualquier aplicación en Android son los siguientes [41]:

Activity El pilar de cualquier aplicación se denomina *Activity*, y se implementa en Android mediante la clase del mismo nombre `Activity` del paquete

`android.app`. Se puede pensar en una actividad como el equivalente en Android a una ventana en cualquier software para ordenador. Si bien es posible que una actividad no tenga una interfaz de usuario, lo más habitual en estos casos es crear componentes de tipo *Content Provider* o *Service*, de los que se habla a continuación.

Content Provider Un componente *Content Provider* dota de un cierto nivel de abstracción a cualquier tipo de datos almacenados en el dispositivo que pueden ser accedidos por varias aplicaciones. El propio modelo de desarrollo para Android anima a los desarrolladores a hacer sus datos disponibles para el resto de aplicaciones. Mediante un *Content Provider* es posible lograrlo, al mismo tiempo que se conserva un control total sobre la manera en que se accede a estos datos. En Android, las clases con las que implementar este tipo de componentes se encuentra en el paquete `android.provider`.

Intent Un componente *Intent* es una comunicación interna que notifica a las aplicaciones acerca de diferentes eventos que tienen lugar en el dispositivo, desde cambios en el estado hardware — cuando, por ejemplo, se inserta o extrae una tarjeta SD — a datos entrantes — ser recibe un SMS — o eventos producidos por las propias aplicaciones — un nuevo componente *Activity* ha sido lanzado —. Las aplicaciones no sólo pueden responder a un *Intent*, sino que es posible implementar *Intents* propios para conocer cuándo se suceden ciertas situaciones, gracias a las clases incluidas en el paquete `android.content.intent`.

Service Por último existen los componentes tipo *Service*. Mientras que los componentes de tipo *Activity*, *Content Provider* e *Intent* suelen tener un tiempo de ejecución determinado y pueden ser finalizados en cualquier momento, un componente *Service* está diseñado para ejecutarse de manera indefinida, incluso desvinculado de cualquier otro tipo de actividad. Aplicaciones típicas de este componente pueden ser, por ejemplo, un componente *Service* que compruebe actualizaciones en el correo electrónico cada cierto tiempo, o que continue reproduciendo música a pesar de que el componente *Activity* del reproductor ya no se encuentre en ejecución. Un componente *Service* se implementa mediante la clase *Service* del paquete `android.app`.

Ciclo de vida de las aplicaciones

En Android cada componente se ejecuta en su propio proceso y tiene un ciclo de vida definido [42]. Una actividad tiene tres posibles estados, y existe una serie de métodos que notifican acerca del paso de uno a otro. La figura 4.6 muestra de manera gráfica el ciclo de vida completo de un componente y las transiciones entre estados que pueden ocurrir durante su vida.

Como se ha comentado previamente, una actividad tiene tres estados:

- Está *en ejecución* o *activa* si está en el primer plano de la pantalla , es decir, si es visible para el usuario y es el foco de sus acciones. En este estado, la actividad se encuentra en la cima de la pila de actividades.

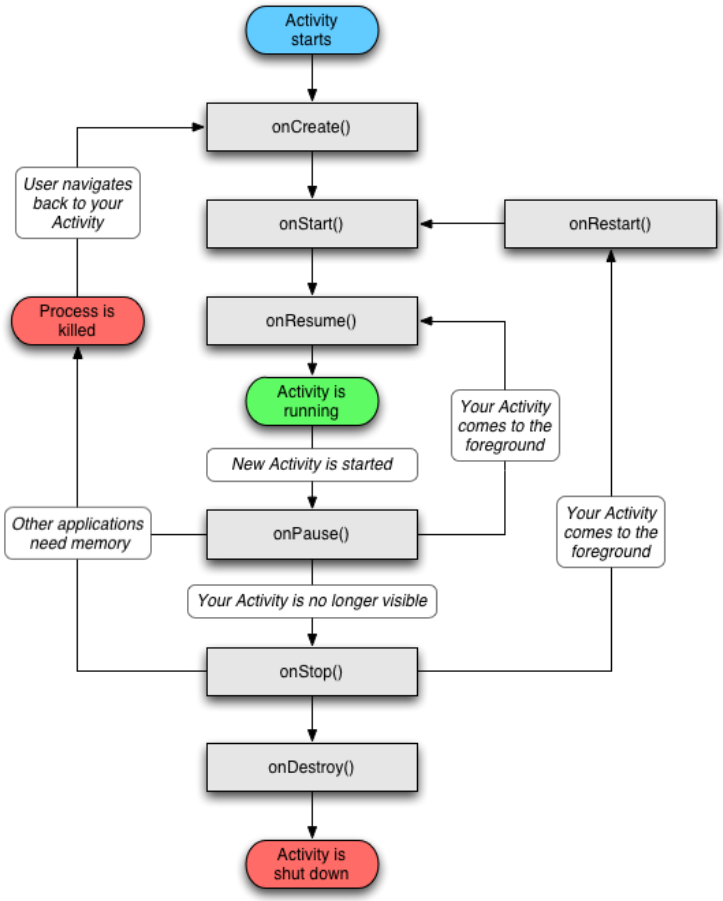

Figura 4.6: Ciclo de vida de una aplicación de Android

- Está *pausada* si no se encuentra en primer plano pero aún resulta visible para el usuario, es decir, otra actividad está por encima de ella pero o bien es transparente o no cubre toda la pantalla. Bajo condiciones extremas de falta de memoria, el sistema operativo puede decidir matar el proceso.
- Esta *parada* si ha perdido completamente la visibilidad por parte del usuario. Aún guarda su estado e información, pero su proceso será matado por el sistema si se necesita memoria para otros procesos.

Cuando se necesita memoria para otros procesos y la actividad está *pausada* o *parada*, el sistema operativo puede pedirle que finalice su ejecución mediante una llamada a `finish()`, o simplemente matando el proceso.

Cada vez que una actividad cambia su estado, es notificada mediante la llamada a un determinado método. A continuación se enumeran estos métodos y se localizan dentro del ciclo de vida de una actividad:

- El método `onCreate()` se llama al crear la actividad por primera vez. Es entonces cuando se deben crear las interfaces y todo el proceso de inicialización de la aplicación. Todas las actividades deben obligatoriamente implementar este método.
- Cuando la actividad ha sido *parada* y vuelve a estar *activa*, se llama al método `onRestart()`.
- Cuando la actividad se muestra por primera vez al usuario, se llama al método `onStart()`.
- El método `onResume()` se llama justo antes de que la actividad comience a interactuar con el usuario. En este punto la actividad pasa a estar en la cima de la pila de actividades.
- El método `onPause()` es llamado en el momento en que otra actividad pasa a ser la primera en la pila de actividades. Normalmente suele estar implementado en todas las aplicaciones, pues la actividad pasa a poder ser matada en cualquier momento y debe guardarse su información de manera persistente.
- En el momento en que una actividad deja de ser visible para el usuario y pasa a estar *parada*, se llama al método `onStop()`. Esto puede ocurrir cuando la actividad está siendo destruida u otra actividad está siendo restaurada.
- Finalmente, el método `onDestroy()` marca el final del ciclo de vida de una actividad, antes de ser destruida.

Estructura y contenido de un proyecto Android

Cualquier proyecto para Android se organiza en torno a una estructura de directorios similar a la de cualquier otro proyecto Java. La estructura básica

definida al crear un nuevo proyecto Android, bien mediante la herramienta android incluida en el SDK, o bien mediante un IDE con soporte para Android, se presenta a continuación:

- El fichero `AndroidManifest.xml` es uno de los elementos más importantes, ya que describe la aplicación y cada uno de sus componentes — *Activity*, *Content Provider*, *Intent* o *Service* —, ay define los permisos de seguridad y librerías de las que se hará uso, la versión mínima de Android requerida o aquella para la que la aplicación ha sido específicamente diseñada.
- El archivo `build.xml` es un script de Ant [43] para compilar la aplicación e instalarla en el dispositivo.
- El fichero `default.properties` contiene propiedades usadas por el script de Ant.
- La carpeta `bin/` contiene la aplicación, una vez compilada.
- La carpeta `libs/` incluye cualquier librería externa que la aplicación pueda requerir.
- La carpeta `src/` contiene el código fuente de la aplicación.
- En el directorio `res/` se guardan otros recursos estáticos — iconos, imágenes — que se incluirán en la aplicación compilada.

La primera vez que se compila una aplicación, Android genera automática la clase R.java, que contiene una serie de constantes vinculadas a los diferentes recursos almacenados en la carpeta `res/` y gracias a las cuales es posible referenciarlos desde el código fuente. Esta clase se actualiza automáticamente y no debe modificarse a mano.

Como se ha comentado previamente, dentro de la carpeta `res/` se incluyen los diferentes recursos que la aplicación necesita. Dada su importancia, merece la pena detenerse en describir brevemente algunos contenidos más habituales que se encontrarán bajo este directorio:

- Las imágenes se guardan en la carpeta `/res/drawable`.
- La carpeta `res/layout/` contiene las interfaces de usuario definidas mediante XML, algo de lo que se hablará más adelante en la sección 4.2.1.
- La carpeta `res/menu/` contiene los archivos XML que definen las especificaciones de los menús de la aplicación. Este punto se cubrirá en la sección 4.2.2.
- La carpeta `res/values` almacena los archivos XML donde se definen los *strings* o textos usados en la aplicación. Es posible crear diferentes subcarpetas para los diferentes idiomas soportados por la aplicación.

4.2. Desarrollo e implementación

En esta sección se describirá en detalle los pasos más importantes seguidos a la hora de implementar la aplicación *WPS*, una vez todas las decisiones acerca de su diseño han sido tomadas.

4.2.1. Construcción de la interfaz principal

A los elementos de cualquier interfaz de usuario en Android se les da el nombre en inglés de *widgets*. Existe una gran cantidad de *widgets* distintos que pueden ser usados en una aplicación, como *layouts* — usados para definir la disposición del resto de *widgets* en pantalla — botones, textos o imágenes. Todos ellos heredan de la clase `View` del paquete `android.view` del SDK de Android, por lo cualquier desarrollador tiene también la posibilidad de crear sus propios widgets para incluirlos en sus aplicaciones.

Aunque técnicamente es posible añadir todos los *widgets* directamente a través del código fuente de la aplicación, la manera más usual es usar un archivo XML. Estos archivos XML especifican la relación entre los diferentes *widgets*, y como se describió en la sección 4.1.5, Android los considera un recurso más de la aplicación.

Cada archivo XML presenta una estructura en forma de árbol, en la que cada elemento especifica uno de los *widgets* que conforman la interfaz. Los atributos de cada elemento son en realidad las propiedades del mismo, Por ejemplo, en un *widget* de tipo `Button` es posible utilizar `android:textStyle=«bold»` para mostrar el texto que incluye en negrita.

El SDK de Android incluye una herramienta llamada `aapt` que es la encargada de manejar estos archivos XML. Entre sus funciones está la de generar la clase `R.java` que vincula los elementos definidos en los archivos XML con el propio código en Java, siendo así posible acceder a ellos.

Dado que la aplicación *WPS* consta de un único componente de tipo *Activity*, sólo ha sido necesario definir una interfaz. El elemento principal y que ocupa la mayor parte de la pantalla es el mapa donde el usuario podrá ver su ubicación actual. Se detallará el uso del servicio Google Maps en la sección 4.2.3. Además, en la parte superior de la pantalla se han añadido tres campos de texto para mostrar diferente información acerca de la posición geográfica actual y la red inalámbrica a la que se corresponde, el estado del GPS y, en caso de estar activado, la posición que éste reporta y el error cometido por la aplicación.

De esta manera, el aspecto que presenta el archivo `main.xml` de la aplicación *WPS* se puede observar en la muestra de código 6.

Una vez definida la interfaz de usuario y almacenada en el fichero `main.xml` dentro de la carpeta `res/layout/`, es necesario *conectarla* con el código Java de la aplicación. Para ello, sólo es necesario hacer referencia a la clase `R.java` a través de una llamada a `setContentView()` dentro del método `onCreate()` de la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView android:id="@+id/wifi_position_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/gps_position_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/error_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <com.google.android.maps.MapView
        android:id="@+id/main_map"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:enabled="true"
        android:apiKey="0OWNmRt_9moaIuObu9RLXqHlQP6Ug7kTGvhoaeA" />
</LinearLayout>
```

Listing 6: Contenido del fichero `main.xml`

Además, para acceder a los diferentes *widgets* definidos, se utiliza el método `findViewById()`. Cada uno de los *widgets* que se desee referenciar desde el código Java debe haber sido previamente identificado en el correspondiente fichero XML. Como se puede observar en la muestra de código 6, cada elemento incluye un atributo `android:id`. Esto genera una nueva constante en la clase `R.java`, que posteriormente puede ser accedida desde otras clases de la aplicación.

En la muestra de código 7 se expone parte del código referente a la creación de la interfaz y el acceso a sus elementos.

4.2.2. Construcción del menú de opciones

Android soporta el uso de menús en las aplicaciones. La mayoría de dispositivos disponen de un botón físico para mostrarlo, mientras que otros ofrecen mecanismos alternativos con el mismo fin.

El menú opera en dos modos, gestionados por el propio sistema operativo. Cuando el usuario pulsa el botón para desplegar el menú, se muestran las seis primeras opciones del menú como botones grandes y con un icono identificativo, en caso de haberse definido. En caso de que el menú disponga de más de seis opciones, el sexto botón pasa a servir como acceso al segundo modo del menú, en el que el resto de opciones no visibles en un primer momento son presentadas como una lista desplazable.

```
public class WPSApp extends MapActivity{

    private TextView textoWifi;
    private TextView textoGps;
    private TextView textoError;
    private MapView mapa;

    @Override
    public void onCreate(Bundle savedInstanceState){
        setContentView(R.layout.main);
        textoWifi = (TextView)
            findViewById(R.id.wifi_position_text);
        textoGps = (TextView)
            findViewById(R.id.gps_position_text);
        textoError = (TextView)
            findViewById(R.id.error_text);
        mapa = (MapView) findViewById(R.id.main_map);
        /*...*/
    }
}
```

Listing 7: Acceso a la interfaz `main.xml` desde el código Java

En lugar de crear el menú en el método `onCreate()`, es necesario implementar `onCreateOptionsMenu()` definido en la clase `Activity`. Este método recibe un objeto de tipo `Menu`, sobre el cual se añaden las opciones que se consideren oportunas. Cada una de estas opciones está representada por un objeto de tipo `MenuItem`, a través del cual se pueden definir todas sus propiedades, como el icono o el texto mostrados.

A la hora de añadir las opciones al menú, al igual que para el diseño de cualquier interfaz en Android, existen dos alternativas. Por un lado, la opción *artesanal* implica la creación manual de los objetos `MenuItem` mediante llamadas al método `add()` de la clase `Menu`.

La otra opción saca provecho de la definición de interfaces mediante XML. Mediante un objeto de la clase `MenuInflater` es posible instanciar los objetos de tipo `Menu` a partir del archivo XML donde se describen sus componentes. De este modo, una vez definida la estructura del menú en el archivo XML correspondiente, la creación del menú de opciones resulta mucho más sencilla, tal y como se puede comprobar en la muestra de código 8, además de mejorar la reusabilidad de la aplicación, pues únicamente es necesario modificar el archivo XML del menú para cambiar el aspecto del mismo.

En la aplicación *WPS* las opciones incluídas en el archivo XML del menú y por tanto disponibles a través del menú son las siguientes:

- *Wi-Fi Scan ON/OFF*: Permite activar y desactivar el scanner Wi-Fi.

```java
@Override
public boolean onCreateOptionsMenu(Menu menu){
    MenuInflater mi = getMenuInflater();
        mi.inflate(R.menu.options_menu, menu);
}
```

Listing 8: Creación del menú de opciones en Android

- *GPS ON/OFF*: Permite activar y desactivar el dispositivo GPS.
- *Wi-Fi Scan Times*: Permite seleccionar la frecuencia, en segundos, con las que se escanean las redes Wi-Fi.
- *Quit*: Detiene la aplicación y libera los recursos usados. Los dispositivos Wi-Fi y GPS quedan activados o desactivados en función de su estado previo a la ejecución de la aplicación.

En el caso de la opción «*Wi-Fi Scan Times*», implica la creación de un submenú, pues una vez seleccionada esta opción debe desplegarse otro menú donde poder seleccionar el intervalo en segundos entre escaneo y escaneo. La definición del submenú en el archivo XML se consigue mediante la definición de subelementos a los que se les imprime cierto comportamiento, en este caso es posible activar una y sólo una de las tres opciones disponibles.

La definición de la interfaz del menú en XML sigue las mismas pautas que las de los archivos XML de las interfaces de la aplicación. El fichero XML `menu.xml` de la carpeta `res/menu/` que define la estructura del menú de opciones tiene el siguiente aspecto:

```xml
<menu>
    <item android:id="@+menu_id/scan"/>
    <item android:id="@+menu_id/gps"/>
    <item android:id="@+menu_id/change_scan_times">
    <menu>
        <group android:checkableBehavior="single">
            <item android:id="@+menu_id/change_scan_times_1"/>
            <item android:id="@+menu_id/change_scan_times_2"/>
            <item android:id="@+menu_id/change_scan_times_3"/>
        </group>
    </menu>
    </item>
    <item android:id="@+menu_id/quit"/>
</menu>
```

Listing 9: Definición mediante XML de la estructura del menú de opciones

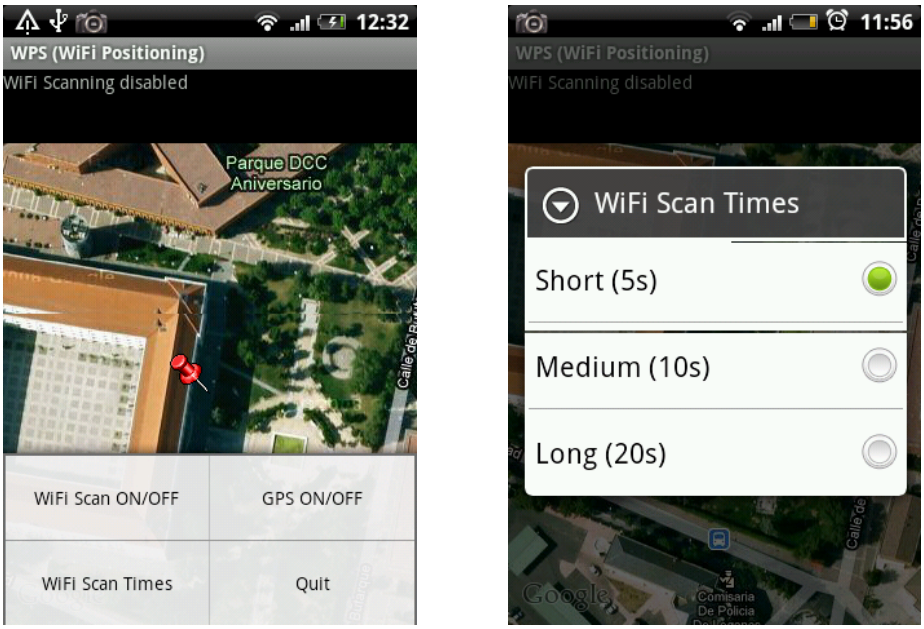

Figura 4.7: Presentación del menú de opciones

De esta manera, cuando el usuario de la aplicación pulsa el botón «Menú», el menú de la aplicación se despliega presentando el aspecto mostrado en la figura 4.7.

Una vez creado el menú, la aplicación permanece a la espera de notificaciones cada vez que el usuario seleccione una de las opciones. Estas notificaciones son recibidas por el método `onOptionsItemSelected()`, definido también en la propia clase `Activity`. Dado que cada una de las opciones ha sido identificada en el archivo `menu.xml` mediante el atributo `android:id`, un simple `switch` basta para determinar cuál ha sido la seleccionada y actuar en consecuencia. El método `onOptionsItemSelected()` se usa sin tener en cuenta los diferentes modos de operación del menú. La muestra de código 10 muestra cómo está implementado este método en la aplicación *WPS*.

4.2.3. Uso de Google Maps

Uno de los servicios más populares de Google es Google Maps, y en Android este servicio se encuentra ya integrado y listo para usarse a través del paquete `com.google.android.maps`. Este paquete proporciona clases para crear aplicaciones que vayan a hacer uso de los mapas — `MapActivity` —, mostrarlos — `MapView` —, manejarlos — `MapController` —, dibujar y representar objetos en ellos — `Overlay` — o gestionar y operar con puntos geográficos — `GeoPoint` —, entre otras. Además, es posible vincular este servicio a los servicios de localización de Android, de manera que se muestre dónde se encuentra el dispositivo.

```
@Override
public boolean onOptionsItemSelected(MenuItem item){
    switch (item.getItemId()){
        case R.menu_id.scan:{
            /*...*/ break;
        }
        case R.menu_id.gps:{
            /*...*/ break;
        }
        case R.menu_id.change_scan_times_1:{
            /*...*/ break;
        }
        case R.menu_id.change_scan_times_2:{
            /*...*/ break;
        }
        case R.menu_id.change_scan_times_3:{
            /*...*/ break;
        }
        case R.menu_id.quit:{
            /*...*/ break;
        }
    }
    return false;
}
```

Listing 10: Gestión del menú de opciones en Android

Integrar Google Maps en una aplicación para Android es un proceso sencillo, si bien requiere de la aceptación de ciertos términos legales y condiciones de uso. Es importante recalcar que existen otras alternativas libres y gratuitas, como por ejemplo OpenStreetMap [44], cuyo uso también es posible y está soportado por Android.

El primer paso para el uso de Google Maps en una aplicación es obtener una clave denominada *API Key*, que da derecho a utilizar los servicios. Para ello se debe estar registrado en Google Maps, para lo que a su vez es necesario disponer de una cuenta de Google activa. Una vez cumplidos estos requisitos, los pasos son los siguientes:

- Obtener el resumen MD5 del certificado con el que firmaremos la aplicación, descrito en detalle a continucación.
- Registrar el resumen MD5 en la página Web de registro de Google Maps [45], donde se deben aceptar las condiciones de uso.
- Incluir la *API Key* generada en el código de la aplicación. Esta *API Key* es exclusiva para las aplicaciones firmadas con el certificado del que se obtuvo el MD5.

Todas las aplicaciones en Android deben firmarse usando un certificado, de manera que se vincule la aplicación con su desarrollador. Tras la instalación del SDK se genera automáticamente un certificado *debug* que se puede usar durante el desarrollo, si bien es necesario generar un certificado personal a la hora de publicar y distribuir las aplicaciones, bien a través de Android Market o cualquier otro medio. Para obtener el resumen MD5 de un certificado se usa la herramienta keytool proporcionada por Java [46].

Una vez se ha obtenido la *API Key*, la manera de incluir un mapa en una aplicación es haciendo que su clase principal — `WPSApp` — herede de la clase `MapActivity` incluida en el paquete `com.google.android.maps` en lugar del habitual `Activity`, e incluir en ella un objeto de tipo `MapView`.

La *API Key* por su parte debe incluirse como un atributo más en la definición del objeto de tipo `MapView` en el archivo XML cuyos elementos conforman la interfaz de la aplicación, tal y como se muestra a continuación:

```
<LinearLayout>
    <!-- ... -->
    <com.google.android.maps.MapView
        android:id="@+id/main_map"
        android:apiKey="0OWNmRt_9moaIuObu9RLXqHlQP6Ug7kTGvhoaeA"
    />
    <!-- ... -->
</LinearLayout>
```

Listing 11: Detalle de la *API Key* incluída en la definición de la interfaz gráfica

4.2.4. Dibujar elementos en el mapa

Al igual que en la versión Web de Google Maps, en Android también es posible añadir capas encima del propio mapa en la que poder marcar puntos de interés o dibujar cualquier elemento, mediante la creación de una clase que herede de la clase `Overlay` del paquete `com.google.android.maps` e implementando el método `draw()`.

En el caso de *WPS*, la mayor parte de la interfaz consta de un mapa donde el usuario necesita visualizar de algún modo su posición. Por tanto es necesario añadir una capa al mapa para realizar esta función, algo de lo que se encarga la clase `CustomLocationOverlay`. En concreto, su cometido es el de dibujar la chincheta en las coordenadas correspondientes a la ubicación actual del usuario.

Es importante recalcar que todos los elementos dibujados en el mapa se mueven de manera coherente con el mismo, algo de lo que se encarga la propia clase `Overlay`, de manera que el usuario puede desplazar el mapa o hacer zoom en él sin que el aspecto de la capa se vea alterado.

Una vez se ha implementado la clase `CustomLocationOverlay`, la nueva capa para el mapa ya está definida, siendo necesario añadirla a la vista del mismo, es

decir, al objeto de tipo `MapView` de la clase principal `WPSApp`. Por supuesto, un mismo `MapView` puede tener asociados varios objetos de tipo `Overlay`, aunque en este caso la aplicación sólo necesita uno.

La muestra de código 12 incluye la implementación completa de la capa CustomLocationOverlay utilizada en la aplicación *WPS*.

```
private class CustomLocationOverlay extends Overlay{
    @Override
    public boolean draw(Canvas canvas, MapView mapView,
        boolean shadow, long when){
        super.draw(canvas, mapView, shadow);

        Point screenPts = new Point();
        mapView.getProjection().toPixels(myGeoPoint, screenPts);

        Bitmap bmp = BitmapFactory
            .decodeResource(getResources(), R.drawable.red_pushpin);
        canvas.drawBitmap(bmp, screenPts.x-32, screenPts.y-32, null);
        return true;
    }
}
```

Listing 12: Definición de la capa `CustomLocationOverlay`

Cabe reseñar que el desarrollador debe ser especialmente cuidadoso a la hora de *colocar* una imagen en el mapa. El punto de referencia que toma como parámetro el método `drawBitmap()`, se correponde con la esquina superior izquierda de la imagen que se desee dibujar. Dado que el punto que la aplicación toma como referencia es la localización exacta del usuario, y que la imagen utilizada señala un punto en la esquina inferior derecha, es necesario desplazar la imagen para hacer coincidir ambos puntos y que la imagen señale el punto exacto donde el usuario está ubicado.

De este modo, la interfaz de la aplicación ya se encuentra totalmente definida, tras los pasos descritos en las secciones anteriores. La figura 4.8 muestra el resultado de añadir la nueva capa a la vista habitual del mapa de la aplicación.

4.2.5. Gestión de la señal Wi-Fi

La clase `WifiManager` del paquete `android.net.wifi` es la encargada de manejar todos los aspectos relacionados con la conectividad Wi-Fi, entre ellos:

- La lista de redes inalámbricas configuradas, que puede ser consultada y editada en cualquier momento.
- La conexión a una red Wi-Fi activa, si existe. Es posible establecer o romper una conexión, así como obtener información acerca de su estado.
- La definición de componentes *Intent* que son enviados para notificar eventos relacionados con el estado de la conectividad Wi-Fi.

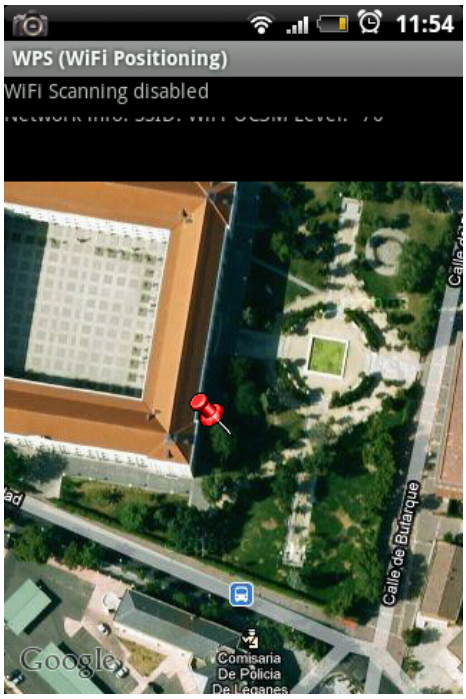

Figura 4.8: Interfaz de la aplicación *WPS*

- Los resultados de los escaneos en busca de puntos de acceso, que incluyen información completa de cada uno de ellos.

Este último punto será el que resulte de mayor interés para los objetivos del proyecto, pues para determinar la localización del usuario será necesario recoger información acerca de las redes inalámbricas al alcance del dispositivo, como se verá más adelante en la sección 4.2.8.

Para conseguir una instancia de la clase `WifiManager`, es necesario incluir la llamada a `getSystemService(Context.WIFI_SERVICE)`.

La aplicación debe ser notificada cuando se lleve a cabo un escaneo de los puntos de acceso disponibles. Como se decribió en la sección 4.1.5, un componente *Intent* es un mensaje interno que se utiliza para notificar a las aplicaciones acerca de diferentes eventos que tienen lugar en el dispositivo. La clase `WifiManager` define el *Intent* `SCAN_RESULTS_AVAILABLE_ACTION`, que informa del momento en que un escaneo ha finalizado con éxito y los resultados se encuentran disponibles.

Por otro lado, la clase `BroadcastReceiver` representa un objeto que es capaz de recibir los *Intents* para los que esté *registrado* o *suscrito*, y ejecutar una acción a través del método `onReceive()`.

Dado que existen multitud de *Intents* definidos, se utiliza un objeto de la clase `IntentFilter` para crear un filtro y actuar exclusivamente cuando se

reciba un tipo de *Intent* determinado.

Por tanto, combinando estos tres objetos es posible *registrar* la aplicación para ser notificada por los *Intent* de tipo `SCAN_RESULTS_AVAILABLE_ACTION` y actuar de la manera oportuna. La muestra de código 13 ayuda a comprender este proceso.

```
public boolean start_wifi(){
    wifiManager = getSystemService(Context.WIFI_SERVICE);
    IntentFilter i = new IntentFilter();
    i.addAction(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION);
    registerReceiver(wifiEvent, i);
}

private BroadcastReceiver wifiEvent = new BroadcastReceiver(){
    @Override
    public void onReceive(Context arg0, Intent arg1){
        /*...*/
    }
};
```

Listing 13: Gestión del dispositivo Wi-Fi

4.2.6. Gestión de la señal GPS

WPSApp puede hacer uso de la señal GPS para obtener una medida fiable del error cometido al calcular la posición geográfica del usuario. Su uso es opcional y se ha incluído principalmente para poder obtener un valor cuantitativo con el cual evaluar la fiabilidad de la aplicación.

Todas las clases relacionadas con el control del GPS del dispositivo están contenidas en el paquete `android.location`. Las clases más importantes y con las que la aplicación ha de interactuar son:

- `Location`: representa una posición geográfica.
- `LocationListener`: captura los eventos asociados al dispositivo de localización.
- `LocationManager`: gestiona el dispositivo de localización.

Estas clases no están relacionadas exclusivamente con el dispositivo GPS sino también con los otros sistemas que tiene Android para determinar la ubicación del usuario, como por ejemplo la información de la red a la que está conectado el terminal.

La clase `LocationManager` es la encargada de gestionar el dispositivo GPS. Al igual que en el caso de la tarjeta inalámbrica del dispositivo, para obtener

una instancia de la clase es necesario incluir en la aplicación una llamada a `getSystemService(LOCATION_SERVICE)`. Nótese el cambio de parámetro, pues el servicio al que se pretende tener acceso es en esta ocasión diferente.

Una vez activado el uso del GPS, la aplicación debe ser notificada de alguna manera cada vez que la posición geográfica proporcionada por el GPS varíe. La manera en que se consigue es muy similar a la presentada en el apartado anterior para el dispositivo Wi-Fi.

El primer paso es solicitar al dispositivo GPS actualizaciones en la localización del dispositivo móvil. Esto se consigue mediante una llamada al método `requestLocationUpdates()`, que recibe cuatro parámetros:

- El proveedor del que se desea recibir notificaciones. Como se ha comentado previamente, existen diferentes sistemas para determinar la localización del dispositivo móvil. El proveedor que se quiere usar en este caso es el GPS, por lo que se usará el valor `GPS_PROVIDER`
- El tiempo mínimo, en milisegundos, que debe transcurrir entre actualizaciones.
- La distancia mínima, en metros, entre actualizaciones.
- El objeto de tipo `LocationListener` que recibirá las actualizaciones.

Como se ha visto, la interfaz `LocationListener` es la encargada de capturar los eventos que genera el dispositivo GPS. Entre los métodos que define se encuentran `onProviderDisabled()`, `onProviderDisabled()`, `onStatusChanged()` y `onLocationChanged()`, cada uno de ellos diseñado para ser llamado en respuesta a cierto tipo de eventos — *Intents* —.

Por tanto, el segundo paso implicar definir en la clase `WPSApp` un objeto de tipo `LocationListener` que reciba las actualizaciones del dispositivo GPS a través del método `onLocationChanged()`. Este método sencillamente calculará la distancia entre los puntos geográficos proporcionados por la aplicación y el GPS, y actualizará la información en pantalla del error cometido.

La muestra de código 14 ilustra acerca del uso del GPS en la aplicación.

Por último cabe reseñar que, a diferencia de la tarjeta inalámbrica y por razones de privacidad, Android no permite que una aplicación active o desactive el dispositivo GPS por sí sola, por lo que si éste no se encuentra ya activado, es necesario emplazar al usuario a activarlo él mismo a través de las opciones de su dispositivo.

4.2.7. Gestión de la rotación y orientación del dispositivo

Suele ser habitual que los dispositivos móviles con Android instalado incluyan entre sus características un acelerómetro para determinar la rotación del dispositivo, algo que resulta de gran utilidad en ciertas aplicaciones. Por tanto

```
public boolean start_gps(){
    location_manager = getSystemService(LOCATION_SERVICE);
    location_manager.requestLocationUpdates
        (LocationManager.GPS_PROVIDER,
        Constants.SCAN_WAIT[scanWait], 0, location_listener);
}

private LocationListener location_listener = new LocationListener(){
    @Override
    public void onLocationChanged(Location location){
        /*...*/
    }
}
```

Listing 14: Gestión del dispositivo GPS

es necesario detectar y manejar las situaciones en las que el dispositivo rote, de manera que la interfaz no se vea afectada, algo para lo que Android está preparado y ofrece diferentes alternativas.

Es importante saber que, por defecto, cuando hay cualquier cambio que afecte a la configuración del dispositivo, Android destruye y vuelve a crear todas las actividades almacenadas en memoria. Esto incluye, entre otros, un cambio en la rotación de la pantalla. Perder la información puede suponer un problema para ciertas aplicaciones. En el caso de *WPS*, un cambio en la orientación del dispositivo supone la pérdida de toda la información mostrada en pantalla, tanto textos como mapas, y la parada de los servicios de escaneo de redes Wi-Fi y GPS.

Para evitar perder información en el proceso y ser capaces de retomar una actividad en el punto en el que se encontraba originalmente, es necesario implementar en la actividad alguno de los métodos definidos en la clase *Activity*, tales como `onSavedInstanceState()` o `onRetainNonConfigurationInstance()`. Sin embargo, existe también la alternativa de forzar una cierta actividad a no rotar, ignorando los cambios en la orientación de la pantalla, a través del manifiesto de la aplicación.

Puesto que la rotación del dispositivo no supone ningún cambio en el funcionamiento de la aplicación *WPS*, se ha optado por implementar esta última opción en la actividad principal `WPSApp`, de manera que la interfaz siempre se mostrará en posición vertical, independientemente de la orientación de la pantalla.

Para ello únicamente se debe editar el archivo `AndroidManifest.xml` y añadir el atributo `android:screenOrientation` a la actividad correspondiente, con un valor igual a `portrait` — para orientación vertical — o `landscape` — para orientación horizontal —. Este cambio afecta únicamente a la actividad sobre la que se ha definido el atributo, por lo que en caso de que una aplicación se componga de diferentes actividades, se debe decidir cuáles harán uso de la rotación y cuáles no, editando en el manifiesto todas aquellas que se deseen bloquear.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <application>
        <activity android:screenOrientation="portrait" />
    </application>
</manifest>
```

Listing 15: Restricción de la rotación de pantalla en *WPS*

4.2.8. Geolocalización del usuario

La funcionalidad más importante de la aplicación es aquella que actualiza la localización del usuario periódicamente — el propio usuario puede definir este periodo a través de las opciones, como se explicó en la sección 4.2.2 —. La aproximación que se ha tomado a la hora de determinar la posición geográfica del usuario es la de equiparar ésta con la posición geográfica de la red cuya señal se recibe con mayor potencia, que *presumiblemente* será también la más cercana al usuario.

El proceso completo se puede dividir en diferentes pasos para una mejor comprensión. Cada uno de estos pasos se repiten cada vez que se consume un nuevo periodo:

- Escanear en busca de puntos de acceso detectados por el dispositivo móvil.
- Determinar la red más cercana al usuario, la cual se usará como referencia de su posición.
- Realizar la conexión con el servidor para consultar los datos de dicha red.
- Parsear el documento XML obtenido como respuesta del servidor.
- Actualizar el mapa y la información mostrada en pantalla en base a los datos recibidos.

Escanear los puntos de acceso

Típicamente, el usuario de la aplicación *WPS* estará en constante movimiento. Por tanto es muy probable que las señales de las redes Wi-Fi recibidas por el dispositivo sean diferentes en cada momento, siendo necesario escanear los puntos de acceso detectados periódicamente.

Para la implementación de esta tarea se valoraron dos alternativas, bien mediante un componente *Service*, o bien en un nuevo hilo dentro del propio componente *Activity*.

Por un lado, se planteó la posibilidad de implementar un nuevo componente *Service*, de manera que el proceso se realizase de manera independiente a

la actividad principal, pero la idea fue desestimada pues carece de sentido sin el componente *Activity* asociado. Dicho de otra manera, escanear en busca de redes Wi-Fi para actualizar la localizacón del usuario resulta inútil si el componente *Activity* — `WPSApp` — no se encuentra en ejecución, pues no existe tal localización que actualizar.

Por tanto se decidió implementar la tarea en la propia *Activity*, mediante el uso de objetos de tipo `Runnable` y `Handler`. La clase `Runnable` del paquete `java.lang` representa una acción que puede ser ejecutada, habitualmente en un nuevo hilo. Por su parte, la clase `Handler`, incluida en el paquete `android.os`, sirve para enviar y procesar objetos de tipo `Runnable` asociados a la cola de mensajes de un hilo de ejecución.

De esta manera, es posible planificar los escaneos de la manera deseada, enviando un mensaje con un objeto `Runnable` y fijando su ejecución en el periodo especificado, tal y como se ilustra en la muestra de código 16.

```
private Handler handler = new Handler();
private Runnable scanTask = new Runnable() {
    @Override
    public void run() {
        if(wifiManager.startScan()){
            handler.postDelayed(this, periodo);
        }
    }
};
```

Listing 16: Uso de objetos de tipo `Handler` y `Runnable`

Elección de la red más cercana

Si el escaneo en busca de puntos de acceso se realiza exitosamente, el objeto de tipo `BroadcastReceiver` de la aplicación *WPS* recibirá un componente *Intent* anunciando la disponibilidad de los resultados y se ejecutará automáticamente el método `onReceive()`, tal y como se describió en la sección 4.2.5.

Los datos recogidos por el escaner acerca de las redes Wi-Fi disponibles se pueden obtener gracias al método `getScanResults()` de la clase `WifiManager`, que devuelve una lista de objetos `ScanResult`. Un objeto de `ScanResult` representa un red, y entre la información incluida se incluye su BSSID o dirección MAC, su SSID y el nivel de potencia de la señal recibido.

Tal y como se comentó anteriormente, la localización del usuario será igual a aquella de la red con mayor nivel de potencia recibido. Sólo resta, por tanto, ordenar la lista en función de este campo. Para facilitar esta tarea se ha creado la clase `ScanResultSortByLevel`, que implementa la interfaz `Comparator` y define el criterio de ordenación. Con su uso únicamente es necesaria una llamada al método `sort()` para tener la lista ordenada, si bien es necesario invertir el orden mediante el método `reverse()` para que éste sea el correcto.

Este proceso se ilustra en la muestra de código 17.

```
private BroadcastReceiver wifiEvent = new BroadcastReceiver(){
    @Override
    public void onReceive(Context arg0, Intent arg1) {
        List<ScanResult> results = wifiManager.getScanResults();
        Collections.sort(results, new ScanResultSortByLevel());
        Collections.reverse(results);
        RequestPositionOnline proc =
            new RequestPositionOnline(results);
        proc.run();
    }
};
```

Listing 17: Uso de objetos de tipo `Handler` y `Runnable`

Conexión HTTP

La aplicación hace un uso intensivo de la conexión a internet del dispositivo, dado que la información con la ubicación de las redes Wi-Fi se encuentra almacenada en un servidor. Por tanto, cada vez que la aplicación escanea las redes Wi-Fi al alcance del dispositivo para actualizar la posición geográfica del usuario, necesita hacer una conexión al servidor para descargar los datos necesarios.

Una vez la lista de redes detectadas se encuentra ordenada, se lanza un nuevo proceso — de manera que la interfaz no quede bloqueada hasta que se resuelva — en el que la aplicación conecta con el servidor para consultar las coordenadas del primer punto de acceso en la lista, es decir, aquel cuya señal se recibió con mayor de potencia. De aquí en adelante se conocerá a este punto de acceso como *red canditata*.

Por supuesto, cabe la posibilidad de que la red candidata no se encuentre presente en la base de datos del servidor. En ese caso, se pasará a intentarlo con la siguiente red candidata en la lista, y así sucesivamente hasta que la lista de redes candidatas esté vacía, en cuyo caso se mostrará un mensaje de error al usuario.

A través de la clase `BasicNameValuePair`, es posible definir cada uno de los parámetros como un par de la forma <*clave, valor*>. En este caso, únicamente es necesario un parámetro, esto es, el identificador — el BSSID o dirección MAC — de la red candidata, que se usará a su vez como parámetro en la consulta a la base de datos del servidor.

Todo el intercambio de información entre la aplicación *WPS* y el servidor se realiza bajo el protocolo HTTP. Los datos se encapsulan y se envían a través del método POST, mientras que la respuesta se recibe también en el campo de datos.

La muestra de código 18 ilustra el proceso de conexión con el servidor de la aplicación.

```
List<NameValuePair> values = new ArrayList<NameValuePair>();
values.add(new BasicNameValuePair("param1",candidate.BSSID));

//Creamos la conexion y esperamos una respuesta
HttpPost post = new HttpPost(Constants.URL);
HttpClient client = new DefaultHttpClient();
post.setEntity(new UrlEncodedFormEntity(values));
HttpResponse response = client.execute(post);

//Parseamos el documento XML que devuelve el servidor
parser.parse(response.getEntity().getContent(), xmlhandler);
```

Listing 18: Conexión HTTP en Android

Parsear documento XML

El servidor emite como respuesta un documento XML con las coordenadas de la red solicitada. Este documento presenta el aspecto mostrado en la muestra de código 19.

Con el objetivo de leer estos documentos y extraer la información necesaria incluida en ellos, se ha implementado la clase `XMLHandler`. Se trata de una clase muy sencilla que extiende a la clase `DefaultHandler` incluida en la librería Java SAX. Esta clase permite recorrer de forma completamente automática los elementos de un documento XML, usando varios métodos entre los que destacan `startElement()` y `endElement()`.

De esta manera, tan sólo es necesario identificar los elementos <LAT>y <LON>que definen las coordenadas obtenidas tras la consulta a la base de datos para extraer la información que contienen a través del método `characters()`.

Actualización del estado

Finalmente, una vez conocida la nueva localizacion del usuario, sólo queda actualizar la interfaz de usuario con la información recibida.

Gracias al método `setText()` de la clase `TextView` se actualizan los textos mostrados en pantalla, y la clase `CustomLocationOverlay` se ocupa automáticamente de actualizar la posición de la imagen y dibujarla correctamente en el mapa.

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENTO>
    <PARAMETROS></PARAMETROS>
    <NETWORK>
        <POSITION>
            <LAT>40.41572484329</LAT>
            <LON>-3.5220040171244</LON>
        </POSITION>
    </NETWORK>
</DOCUMENTO>
```

Listing 19: Contenido del documento XML respuesta del servidor

4.2.9. Manifiesto final

Como se ha descrito en la sección 4.1.5, el archivo `AndroidManifest.xml` describe la aplicación y los componentes que proporciona. En esta sección se detallarán los componentes del manifiesto final de la aplicación *WPS*, contenido en el fichero `AndroidManifest.xml` y en que se definen distintos aspectos importantes de la aplicación.

Las principales características de este fichero se enumeran a continuación:

- Definición de la versión de la aplicación y el paquete donde se encuentra (líneas 3 y 4).
- Definición del icono y nombre de la aplicación, así como se habilita el modo *debug* o de depuración (líneas 5 y 6).
- Declaración de uso de la librería `com.google.android.maps`, necesaria para el uso de mapas en la aplicación (línea 7).
- Definición de la primera y única actividad de la aplicación, el nombre de la clase con la que se corresponde y la etiqueta que aparecerá en la parte superior de la pantalla (líneas 8 y 9).
- Restricción de la orientación de la actividad, que sólo actuará en modo *portrait* o vertical (línea 10).
- Definición de la actividad como actividad principal, esto es, la primera en ejecutarse al lanzar la aplicación (línea 12).
- Especificación de que la actividad será mostrada en el Launcher o menú de aplicaciones de Android (línea 13).
- Permiso de localización ordinaria — Wi-Fi, antenas de telefonía — (línea 18).
- Permiso de localización precisa — GPS — (línea 19).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      android:versionCode="2" android:versionName="1.1"
4      package="com.carlosperezb.wps">
5  <application android:icon="@drawable/icon"
6      android:label="@string/app_name" android:debuggable="true">
7      <uses-library android:name="com.google.android.maps"/>
8      <activity android:name=".WPSApp"
9          android:label="@string/app_name"
10         android:screenOrientation="portrait">
11         <intent-filter>
12             <action android:name="android.intent.action.MAIN"/>
13             <category android:name="android.intent.category.LAUNCHER"/>
14         </intent-filter>
15     </activity>
16 </application>
17
18 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
19 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
20 <uses-permission android:name="android.permission.ACCESS_GPS"/>
21 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
22 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
23 <uses-permission android:name="android.permission.INTERNET"/>
24 <uses-permission android:name="android.permission.WAKE_LOCK"/>
25
26 <uses-sdk android:minSdkVersion="7"/>
27 </manifest>
```

Listing 20: Contenido del fichero `AndroidManifest.xml`

- Permiso de uso del GPS (línea 20).
- Permiso de acceso a información de la red (línea 21).
- Permiso de modificación del estado del dispositivo Wi-Fi (línea 22).
- Permiso de acceso a Internet (línea 23).
- Permiso para la prevención de la suspensión del procesador o la pantalla (línea 24).
- Definición de la versión mínima del SDK de Android requerida, la versión 7 del SDK de Android se corresponde con su versión 2.1. (línea 26).

Capítulo 5

Experimentación y Resultados

En este capítulo se describen las diferentes pruebas y experimentos realizados con la aplicación, una vez finalizado su desarrollo, con el fin de asegurar su correcto funcionamiento y obtener unos resultados con los que poder analizar el éxito del proyecto.

5.1. Entorno y procedimiento

Para la correcta realización, es necesario disponer de ciertos elementos:

- Un dispositivo móvil Android con las aplicaciones *Warwalk* y *WPS* instaladas. Este dispositivo debe contar con tarjeta de red inalámbrica y GPS, además de acceso a Internet — bien mediante Wi-Fi o a través de la red de telefonía — para poder establecer una conexión con el servidor web. El dispositivo usado para estos exprimentos ha sido el HTC Hero, que cumple con todos los requisitos mencionados.
- Un entorno con una cierta infraestructura Wi-Fi disponible, donde poder descubrir redes inalámbricas sin dificultad. Las pruebas han tenido lugar en los Campus de Leganés y Getafe de la Universidad Carlos III de Madrid, que suponen un excelente entorno donde poder realizar experimentos, si bien cualquier área urbana tiene por lo general una alta densidad de redes inalámbricas.
- Un servidor web que almacene todos los datos recogidos y donde estén disponibles para su consulta. Para estas pruebas se ha utilizado un servidor propio de la Universidad, en el que están instalados todos los componentes necesarios.

Las pruebas han tenido lugar en dos fases, una por cada aplicación desarrollada. De esta manera, una primera fase implica la recogida de datos de las redes Wi-Fi a través de la aplicación *Warwalk* y la subida de datos al servidor, mientras que en la segunda fase, se prueba la aplicación *WPS* con la que geolocalizar a un usuario mediante redes Wi-Fi, utilizando los datos recogidos en la primera fase.

5.2. Aplicación *Warwalk*

En la primera fase de pruebas, el objetivo principal es recabar información acerca de las redes inalámbricas presentes en el entorno para poder geolocalizarlas. Para ello es necesario el uso de la aplicación *Warwalk*.

Una vez iniciada la aplicación y fijada la posición mediante satélite — lo que puede demorarse en función de diferentes factores —, sólo es necesario caminar para comenzar a descubrir redes inalámbricas y generar datos acerca de sus posiciones. Sin embargo, si se desea obtener los mejores resultados posibles, la mejor opción es caminar rodeando los edificios, pues es en estos casos cuando el algoritmo funciona mejor y se saca el mayor provecho al uso del promedio de las coordenadas en el algoritmo de posicionamiento de la aplicación.

Uno de los principales objetivos en las pruebas es precisamente comprobar el funcionamiento del algoritmo, y la precisión alcanzada a la hora de localizar y distribuir los puntos de acceso, uno de los puntos más delicados e importantes de todo el proyecto.

Gracias al nuevo algoritmo implementado, y al contrario que con otras aplicaciones, el mapa presenta un aspecto más realista, donde los puntos de acceso aparecen distribuidos por la totalidad del mapa y no se muestran alineados a lo largo del recorrido realizado, por lo que se puede afirmar que el sistema de medias cumple uno de sus cometidos.

Respecto a precisión obtenida, resulta prácticamente imposible en este punto conocer de antemano las localizaciones exactas de todos o parte de los cientos de puntos de acceso repartidos por todo el Campus. Será la segunda fase de pruebas utilizando la aplicación *WPS* la que determine en qué medida los puntos de acceso se localizan correctamente.

Las figuras 5.1 y 5.2 muestran los mapas de las redes descubiertas en el Campus de Leganés y Getafe, respectivamente.

En la figura correspondiente al Campus de Leganés se puede observar cierto vacío en la zona del edificio Agustín de Betancourt, en la parte superior izquierda de la imagen, algo que afectará posteriormente a las pruebas realizadas con la aplicación *WPS*. Quizás en este edificio, por su arquitectura, sea en el que más complicado resulta localizar correctamente los puntos de acceso.

La segunda figura, correspondiente al Campus de Getafe, muestra un aspecto más uniforme. Quizás la fisonomía de los edificios en este Campus hace más sencillo localizar correctamente los puntos de acceso, pues resulta sencillo rodear

Figura 5.1: Mapa de redes Wi-Fi descubiertas en el Campus de Leganés

completamente la mayoría de edificios, y éstos son suficientemente estrechos como para que las redes descubiertas en cada lateral sean las mismas. Es en este escenario donde el algoritmo de posicionamiento de la aplicación se comporta de la mejor manera, algo que se verá reafirmado en la segunda fase de las pruebas.

Respecto a la subida de los datos recogidos al servidor del sistema, la operación se llevó a cabo sin ningún tipo de problemas, si bien el tiempo empleado en realizarla resultó excesivo, con el consiguiente aumento en el consumo de batería debido al uso intensivo de la conexión de datos. Hay que tener en cuenta que entre los dos Campus la aplicación Warwalk descubrió mas de 600 puntos de acceso diferentes.

5.3. Aplicación *WPS*

La segunda fase de pruebas

Dado que la aplicación *WPS* no genera ni almacena por sí misma ningún tipo de información acerca de su funcionamiento y rendimiento, durante esta fase se ha utilizado una versión especial que genera un fichero de texto con diferente información recogida durante la ejecución:

- Fecha completa.
- Coordenadas según el dispositivo GPS.
- Coordenadas según la aplicación WPS.

Figura 5.2: Mapa de redes Wi-Fi descubiertas en el Campus de Getafe

- Distancia en metros entre ambos puntos.

Gracias a este fichero ha sido posible generar varios archivos Excel y KML con los que analizar y exponer los datos aquí presentados.

Es importante recalcar que el GPS tampoco es un sistema totalmente preciso. En función del número de satélites captados, entre otros factores, las precisiones obtenidas pueden variar desde unos 2,5 metros hasta 15 metros. Por tanto, la distancia entre las coordenadas del GPS y las de la aplicación *WPS* no se puede considerar como una medida exacta sino como una buena aproximación.

5.3.1. Campus de Leganés

El recorrido por el Campus de Leganés tuvo una duración aproximada de doce minutos, periodo durante el que se recogieron 497 muestras de datos.

La figura 5.3 muestra el error en metros cometido a lo largo del recorrido. A continuación se presenta un resumen con los principales indicadores estadísticos de los datos recogidos:

Error medio (metros):	42.619
Desviación típica (metros):	58.563
Mediana (metros) :	29.071
Error máximo (metros):	410.576
Error mínimo (metros):	0.599

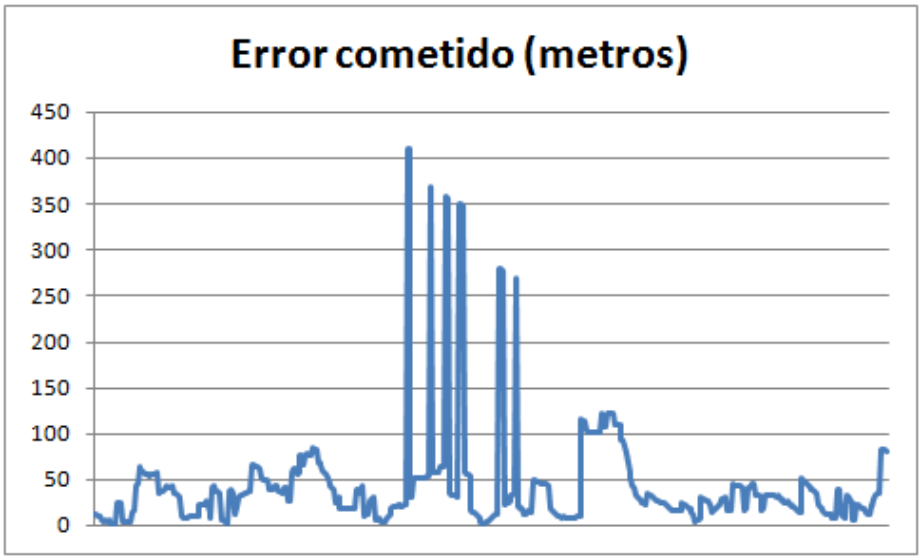

Figura 5.3: Error durante el recorrido por el Campus de Leganés

Se puede comprobar como existen ciertos puntos en los que el error se dispara por encima de los 250 metros y llegando a un máximo de 410 metros. Estos datos suponen únicamente un 3.01 % — 15 muestras — del total — 497 muestras —, por lo que es posible concluir que se trata de datos completamente atípicos que afectan a los resultados obtenidos.

Teniendo en cuenta que la aplicación *WPS* posiciona al usuario únicamente en función de los datos almacenados en el servidor, sin realizar ningún tipo de cálculo, estos datos atípicos pueden achacarse a un error puntual con las coordenadas GPS a la hora de calcular la posición de la red en cuestión.

Por tanto, se decidió eliminar dichas muestras de los datos y realizar de nuevo el análisis. La figura 5.4 muestra la evolución del error a lo largo del recorrido. De estos datos se pueden extraer las siguientes medidas:

Error medio (metros):	33.467
Desviación típica (metros):	26.229
Mediana (metros) :	27.924
Error máximo (metros):	120.774
Error mínimo (metros):	0.599

Los datos más importantes a destacar son el error medio cometido por la aplicación, de 33.467 metros, y la mediana de los datos, que se sitúa en 27.924 metros. Un valor en la mediana inferior al de la media es un buen indicador, pues significa que independientemente de su promedio, el error durante la mitad del recorrido no superó los 27.924 metros.

Se puede ir un poco más lejos en cuanto a los percentiles, pues el 90 % de los errores fueron inferiores a los 65.992 metros, valor que representa casi la

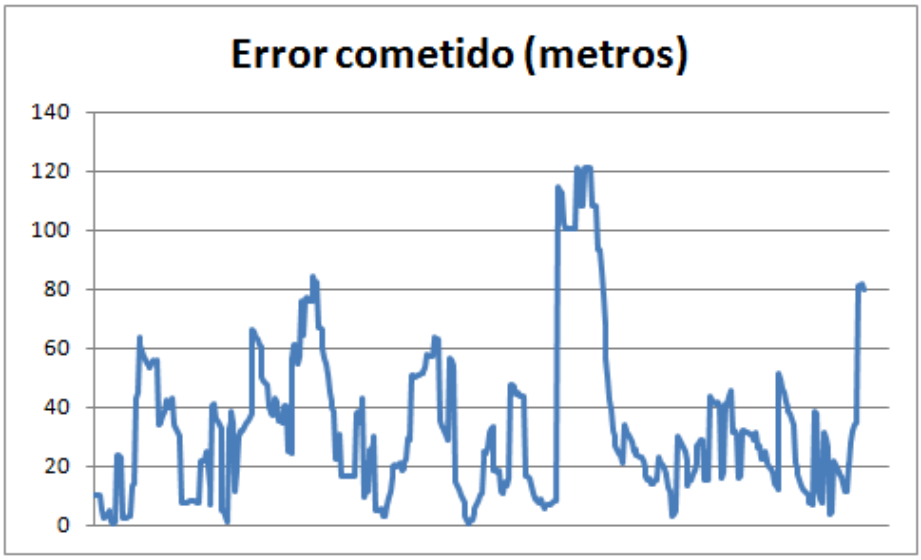

Figura 5.4: Error en el Campus de Leganés, excluyendo datos atípicos

mitad del error máximo observado. Tan sólo existe un 10 % de datos con errores superiores a ese valor, datos que además se concentran en dos tramos concretos del recorrido, durante los minutos cuatro y ocho. Si no se considerase este 10 % de datos, los resultados obtenidos serían los siguientes:

Error medio (metros):	26.626
Desviación típica (metros):	16.412
Mediana (metros) :	24.467
Error máximo (metros):	65.210
Error mínimo (metros):	0.599

Los valores de la media y la mediana de los errores se encuentran más parejos, lo que indica un conjunto de datos más uniforme. Como se comentó en el párrafo anterior, la exclusión del 10 % de muestras tiene un gran impacto en el error máximo, que ve reducido su valor en más de un 45 %, desde 120.774 metros hasta 65,210.

En cualquier caso, el error cometido en ese 10 % de datos sí puede ser achacable a falta de precisión de la aplicación *Warwalk* y no tanto al dispositivo GPS, por lo que a efectos de las conclusiones, se tendrán en cuenta los valores presentados anteriormente y no estos últimos.

Por último, la figura 5.5 presenta en un mapa una comparativa de las coordenadas GPS y las reportadas por la aplicación *WPS*, representadas en color azul y rojo, respectivamente. Aquí se puede comprobar de manera visual cómo el error cometido durante gran parte del recorrido no es grande, y tan sólo en dos tramos alrededor del edificio Agustín de Betancourt, éste se dispara.

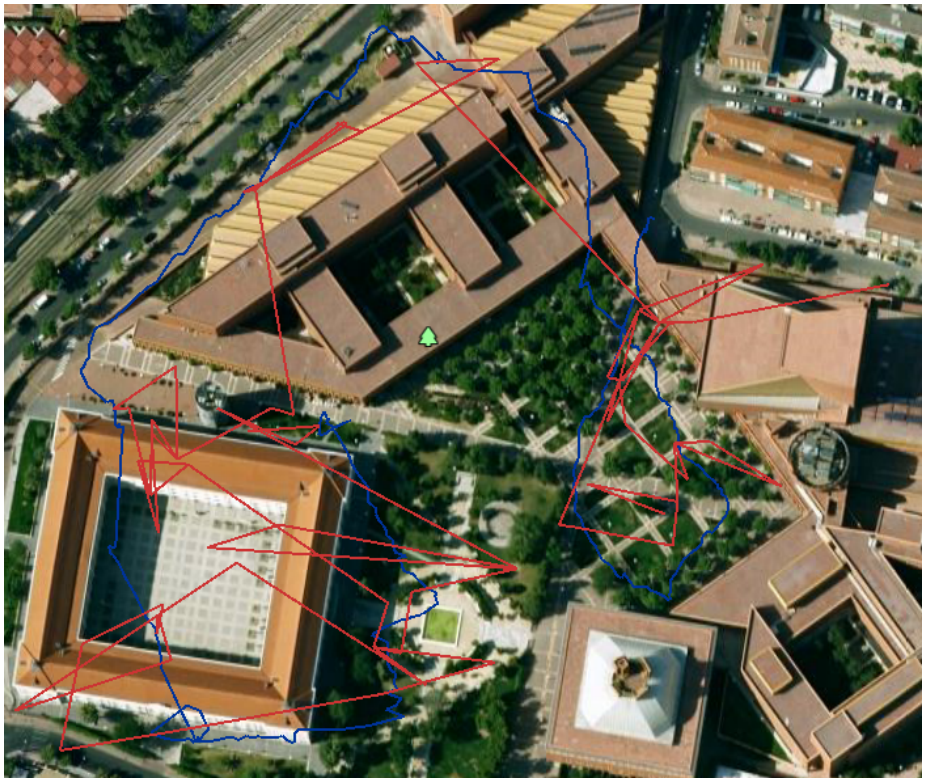

Figura 5.5: Comparativa de rutas en el Campus de Leganés

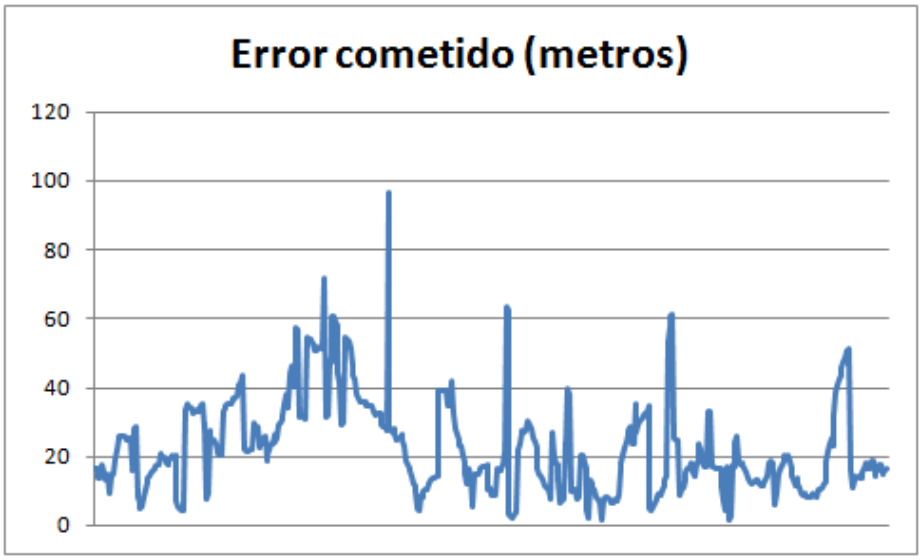

Figura 5.6: Error durante el recorrido por el Campus de Getafe

5.3.2. Campus de Getafe

El recorrido por el Campus de Getafe tuvo una duración aproximada de once minutos, en los que se generaron 530 muestras de datos. La figura 5.6 ilustra cómo fue variando el error cometido en la geolocalización. De los datos recogidos se pueden extraer los siguientes valores acerca del error:

Error medio (metros):	22.458
Desviación típica (metros):	13.725
Mediana (metros) :	18.320
Error máximo (metros):	96.439
Error mínimo (metros):	1.401

Los resultados en este caso son francamente buenos, superando las expectativas. El error medio no es grande — 22.458 metros —, el 50 % de los valores en el error están por debajo de los 18.32 metros, y estos no oscilan demasiado, pues la desviación típica es de 13.725 metros.

Al igual que en el caso del Campus de Leganés, existen algunos valores atípicos. Si bien no son tan exagerados y puedan ser debidos sencillamente a errores cometidos por el propio sistema, cabe destacar que el 90 % de las medidas del error cometido están por debajo de los 40.685 metros, menos de la mitad del error máximo observado.

Los errores más altos se cometen en aquellas zonas más abiertas, como el patio central del Campus, donde a la aplicación *Warwalk* le resulta más complicado posicionar los puntos de acceso.

Figura 5.7: Comparativa de rutas en el Campus de Getafe

La comparación de las rutas por el Campus de Getafe según el dispositivo GPS y la aplicación *WPS* se puede observar en la figura 5.7. Como en el caso del recorrido por el Campus de Leganés, la ruta marcada por el GPS se dibuja en azul, mientras que las posiciones que marca la aplicación *WPS* se muestran en color rojo.

Se puede apreciar como durante el recorrido alrededor del patio central del Campus dos puntos se unen, dando la impresión de que el recorrido es cerrado. En realidad, esto es debido al error cometido por la aplicación en uno de los puntos, en concreto el punto del error máximo durante el recorrido, en el que la aplicación determinó la posición del usuario en un punto por el que se volvería a transitar posteriormente.

En la figura también es posible verificar cómo las coordenadas que proporciona el GPS también tienen su propio error pues, además de fluctuar constantemente, en ocasiones se sitúan en lugares inaccesibles.

En resumen, las pruebas realizadas han buscado analizar el funcionamiento del sistema completo en un entorno real como son dos de los Campus de la Universidad Carlos III de Madrid. Gracias a la aplicación *Warwalk* se ha descubierto y localizado un gran número de redes inalámbricas, cuyos datos

se han subido al servidor del sistema para su uso con la aplicación *WPS*, que ha demostrado ser capaz de determinar la posición geográfica de un usuario con una precisión media ligeramente superior a los 20 metros, lo cual resulta sorprendente teniendo en cuenta el tipo de sistema del que se trata.

En el capítulo siguiente se pueden encontrar las conclusiones extraidas de estas pruebas.

Capítulo 6

Conclusiones y Trabajos Futuros

En este capítulo se extraen las conclusiones finales del proyecto, en base a los resultados obtenidos y los objetivos planteados para el mismo, además de proponer líneas de trabajo futuras. Por otro lado, se ofrece una pequeña visión de lo que ha supuesto el desarrollo del proyecto.

6.1. Gestión del Proyecto

6.1.1. Historia del Proyecto

Una vez aprobados todos los créditos de la titulación de Ingeniería en Informática, se comenzaron a valorar las diferentes opciones para este proyecto de fin de carrera, consultando con algunos de los profesores cuyas asignaturas resultaron más interesantes y atrayentes o con aquellos con los que se había establecido una relación más cercana. Si bien escoger para el proyecto un tema con el que autor disfrute y se sienta cómodo durante su desarrollo resulta esencial, no lo es menos el hecho de contar con un tutor con el que el trato sea agradable y al que poder acudir en momentos de dificultad.

Desde un principio el autor de este proyecto ha tenido interés en la plataforma Android, al tratarse de un sistema abierto, novedoso y de fácil comprensión y aprendizaje, dada la familiaridad con el lenguaje de programación *Java*. Este interés no hizo sino aumentar al adquirir el autor un teléfono móvil con tecnología Android, en concreto el dispositivo HTC Hero, y poder comprobar de primera mano sus ventajas y desventajas. Por tanto se intentó buscar un proyecto relacionado con el desarrollo para esta plataforma.

Respecto a la geolocalización, resultó especialmente interesante el gran número de aplicaciones que un sistema de geolocalización mediante redes Wi-Fi puede tener, al margen de ser una alternativa al uso del GPS. Además el desarrollo

del presente proyecto ofrecía la posibilidad de trabajar conjuntamente con otros antiguos alumnos y desarrollar parte de un proyecto a mayor escala, lo que también se valoró positivamente.

De esta manera, y con la decisión ya tomada, el trabajo en el proyecto comenzó en Marzo de 2010 y finalizó a comienzos del mes de Julio del mismo año. Se trata por tanto de un periodo de desarrollo ligeramente superior a los cuatro meses, exactamente 18 semanas, lo que supone un plazo un tanto ajustado si se tiene en cuenta la duración habitual de este tipo de proyectos. El ritmo constante de trabajo y la ventaja de no tener otras obligaciones académicas hizo posible su finalización dentro del plazo estipulado inicialmente.

Durante el desarrollo del proyecto se pueden distinguir tres fases principales, que han tenido lugar sucesivamente a medida que se ha avanzado en el tiempo y progresado en el trabajo. Estas fases, atendiendo a su orden cronológico, son las siguientes:

- Estudio previo: La primera tarea consistió en la documentación de todos los temas relacionados con el proyecto, desde la instalación del SDK de Android y estudio de las distintas API, hasta la lectura de diferentes trabajos basados en sistemas de geolocalización mediante redes Wi-Fi y técnicas de posicionamiento de puntos de acceso, pasando por la búsqueda y documentación de software y sistemas similares ya disponibles. Esta primera fase abarcó aproximadamente cinco semanas y medio del desarrollo del proyecto.
- Desarrollo del sistema: Una vez se adquirieron los conocimientos básicos necesarios, se pasó a desarrollar el sistema en su totalidad, comprendiendo las fases de diseño, implementación y pruebas. El tiempo empleado en esta fase fue de aproximadamente nueve semanas.
- Redacción de la memoria: El último paso consistió en la documentación del proyecto y la redacción de la presente memoria. Durante la fase de estudio previo se prestó especial atención a la hora de documentar la bibliografía consultada, de manera que parte del trabajo ya estuviese hecho. Esta última fase abarcó las últimas cinco semanas de la totalidad del proyecto, solapándonse la primera de ellas con las últimas tareas en lo referente al desarrollo del sistema.

Por lo general el proyecto ha avanzado constantemente sin que surgiesen demasiadas complicaciones. Los mayores problemas surgieron a la hora de lidiar con aquellos aspectos del proyecto que se alejaban ligeramente de los campos con lo que el lector estaba más familiariazado, tales como las técnicas de posicionamiento de puntos de acceso inalámbricos, donde se requieren conocimientos teóricos y matemáticos más propios de otras ciencias como las telecomunicaciones.

Por otra parte, el entorno de desarrollo en Android resultó muy fácil de aprender y usar, por lo que no se encontró ninguna dificultad en este punto y las aplicaciones se crearon en un corto plazo. La documentación ofrecida por

Google en la página web de Android, así como la del resto de APIs utilizadas durante el desarrollo del proyecto, resultaron de gran ayuda a la hora de obtener información.

Durante el desarrollo de este proyecto, Google publicó la versión 2.2 del SDK de Android, ya disponible para algunos dispositivos. Sin embargo, la imposibilidad de contar con un dispositivo con la nueva versión instalada para probar las aplicaciones hizo que se continuase con el uso de la versión 2.1, que a fecha de la redacción de esta memoria, seguía siendo la más implantada.

6.1.2. Software utilizado

A lo largo del desarrollo de este proyecto se ha utilizado diferente software en cada una de las tareas desarrolladas. A continuación se describen brevemente cada uno de estos programas.

Eclipse

El entorno de desarrollo elegido fue Eclipse [47], no sólo por ser el más recomendado a la hora de desarrollar aplicaciones para Android sino también porque el autor ya estaba familiarizado con él.

El plugin Android Development Tools (ADT) para Eclipse [48] integra todas las herramientas necesarias para el desarrollo para Android usando Eclipse. Entre sus principales ventajas destacan:

- El asistente para la creación de un proyecto Android, con el que crear automáticamente todo el esqueleto necesario para comenzar un nuevo proyecto en Android.
- El emulador de Android y los dispositivos virtuales — Android Virtual Device, AVD — con los que poder depurar las aplicaciones desarrolladas sin necesidad de un dispositivo real, si bien la funcionalidad es limitada, como se debatirá en la sección 6.2.2.
- El asistente para la creación de interfaces visuales, con el que los archivos XML se generan automáticamente.
- La exportación y firma automática de las aplicaciones.

En general, gran parte de las tareas más habituales están automatizadas, lo cual ayuda a un desarrollo más rápido y ágil.

La versión de Eclipse utilizada durante el desarrollo de este proyecto ha sido la versión 3.5.2, junto con la versión 0.9.6 del plugin ADT.

Componentes del servidor

Para la instalación y puesta en marcha del servidor del sistema, se utilizó el siguiente software:

- Apache HTTP Server [49] es un servidor web *open source* que implementa el protocolo HTTP. Se trata del servidor web más usado y soporta el uso de MySQL.
- MySQL [50] ha sido el sistema gestor de bases de datos instalado en el servidor. Está distribuido bajo licencia GNU GPL, siendo por tanto software libre, siempre que su uso no sea comercial.
- La herramienta usada para gestionar la base de datos del servidor ha sido PHPMyAdmin [51]. Se trata de software *open source* escrito en PHP, que permite la creación, modificación y borrado de bases de datos, tablas, campos o filas, ejecutar comandos SQL y gestionar los usuarios y los permisos de las bases de datos. Todas estas operaciones se realizan a través de una interfaz web, con lo que su uso resulta sencillo.
- El último de los componentes utilizados en el servidor es PHP [52], un lenguaje de *scripting* diseñado para la creación de páginas web dinámicas. Su uso en este proyecto se ha limitado a proporcionar un mecanismo para el intercambio de información entre el servidor y la base de datos.

Google Code

Google Code es uno de los sitios web de Google para desarrolladores. Entre los servicios que ofrece se encuentra el de alojamiento de proyectos, que tiene entre sus características un sistema de control de versiones Subversion. Se trata por tanto de una buena herramienta gratuita para alojar el proyecto, llevar a cabo el control de versiones y hacer el código accesible para terceros.

Ambas aplicaciones se encuentra alojadas en Google Code, pudiéndose acceder a las páginas de los proyectos para la aplicación *Warwalk* [54] y la aplicación *WPS* [55], desde donde también es posible descargar los archivos `.apk` de las aplicaciones.

LaTeX

A la hora de redactar la presente memoria, se decidió utilizar LaTeXpor recomendación expresa del tutor del proyecto. LaTeX[56] es un sistema de composición de textos, especialmente orientado a la creación de libros y textos científicos, creado a partir del lenguaje TeX. LaTeXes además software libre, publicado bajo licencia LPPL.

En un principio el entorno resultó totalmente desconocido para el autor. La creación de textos con LaTeXsupone adoptar una filosofía radicalmente diferente

a la que habitualmente se está acostumbrado, ya que está basada en comandos y no es un editor de tipo WYSIWYG, es decir, «lo que ves es lo que obtienes».

Sin embargo, una vez familiarizados con el entorno, resultó más sencillo de lo esperado habituarse al uso de los comandos más frecuentes, con los que finalmente crear un documento de manera sencilla pero cuyos resultados están por encima de los obtenidos con otro tipo de editores de texto.

6.2. Conclusiones

6.2.1. Conclusiones finales

Una vez finalizado el proyecto, es posible hacer balance de los resultados obtenidos y compararlos con los objetivos planteados inicialmente, extrayendo las conclusiones y críticas oportunas.

En este proyecto se ha construido un sistema para el descubrimiento y la geolocalización mediante redes Wi-Fi, utilizando dispositivos con el sistema operativo Android. Usando la primera versión de las dos aplicaciones desarrolladas, es posible recoger los datos necesarios para enviarlos a un servidor desde donde puedan ser consultados posteriormente para la geolocalización de un usuario.

A la vista de los resultados obtenidos, y analizando punto por punto los objetivos planteados para este proyecto, es posible afirmar que éstos se han alcanzado.

Respecto a la recogida de datos, se ha conseguido reutilizar una de las aplicaciones ya disponibles para tal fin para construir una nueva aplicación capaz de descubrir automáticamente redes Wi-Fi y almacenar información acerca de ellas en una base de datos alojada en un servidor web. Se ha implementado un nuevo algoritmo de posicionamiento para las redes e implementado del lado del servidor un sistema de confianza en cada una de ellas, con el fin de mejorar la precisión en la localización y minimizar el impacto de posibles datos erróneos.

Con vistas al futuro, no hay que olvidar el componente colaborativo de esta herramienta. La aplicación puede ser distribuida mediante Android Market u otros canales, y si se consigue que más gente la utilice, sería posible crear una gran base de datos con datos fiables de puntos de acceso de cualquier parte del mundo, abriendo así el camino para el desarrollo de nuevos servicios basados en este sistema.

Por su parte, con la aplicación *WPS* se ha conseguido utilizar los datos almacenados en el servidor para determinar la posición geográfica del usuario. Pese a tratarse de una primera versión del sistema, que utiliza una aproximación al problema bastante simple, se ha alcanzado un nivel de precisión suficiente para su uso en otras aplicaciones, superando incluso la precisión ofrecida por otros sistemas, como el utilizado por Google.

No sólo los objetivos los objetivos principales se han cumplido. Durante el desarrollo del proyecto se han estudiado las principales técnicas de recogida de

datos y posicionamiento de puntos de acceso inalámbricos. Además, el desarrollo de ambas aplicaciones ha permitido conocer las principales características del sistema operativo Android y estudiar su entorno de desarrollo.

Por último, y al margen de los objetivos iniciales, se ha conseguido desarrollar la totalidad del proyecto utilizando exclusivamente software libre y *open source*, con lo que se reduce el coste al mínimo y se hace accesible para todo el mundo, reutilizando también en la medida de lo posible aquellas aplicaciones ya desarrolladas y cuyo uso resultaba factible para los objetivos perseguidos.

6.2.2. Crítica y dificultades

Pese a que los objetivos planteados en el proyecto se han cumplido exitosamente y el balance es positivo, también se quiere aprovechar la ocasión para realizar alguna crítica en base a las dificultades surgidas durante el desarrollo del proyecto.

La fragmentación en Android

Durante el desarrollo del proyecto han surgido diferentes complicaciones que en un principio no estaban previstas. Actualmente Android se encuentra en pleno desarrollo y crecimiento, y esto ha tenido como consecuencia una fragmentación en los terminales que lo incorporan por momentos preocupante. Es posible encontrar dispositivos móviles con Android desde su versión 1.5 hasta la más reciente 2.2, pasando por 1.6, 2.1 o las menos comunes 2.0 y 2.0.1. Por si esto fuera poco, ya está anunciada la versión 3.0 para finales del 2010. Cada versión introduce en su SDK nuevas características y modificaciones a las funcionalidades ya ofrecidas, por lo que no todas las aplicaciones funcionan en cualquier versión del sistema operativo, y por consiguiente en cualquier dispositivo.

A esto hay que sumar que no siempre el fabricante y las operadoras ofrecen actualizaciones para todos sus terminales, provocando que éstos queden desactualizados. Por suerte Android es un sistema abierto, y existen numerosos desarrolladores que compilan nuevos firmware o «Custom ROM» con los que tener la posibilidad de mantener los terminales actualizados.

En el caso que nos ocupa, la aplicación wardrive requiere de la versión 1.6 de Android para funcionar, mientras que el firmware oficial ofrecido por HTC en el momento del desarrollo funcionaba con la versión 1.5. La solución fue instalar una de los varios firmwares no oficiales, con la versión 2.1 de Android. Sin embargo se presentan diferentes desventajas, y es que al no tratarse de una publicación oficial, el funcionamiento no siempre es el correcto, tanto a nivel de software como de hardware. En especial se han encontrado problemas a la hora de recibir la señal del GPS, aunque con las versiones más recientes esos problemas desaparecieron, funcionando todas las características correctamente.

El uso del emulador de Android

Si bien es de agradecer la posibilidad de disponer de un emulador en el que poder simular el funcionamiento de las aplicaciones desarrolladas de antemano y sin necesidad de ningún dispositivo físico, lo cierto es que para aplicaciones como las que se han desarrollado en este proyecto, su uso se limita a comprobar el aspecto de la interfaz, pues el uso tanto del GPS como de la tarjeta inalámbrica son complicados de emular.

Respecto al GPS, existe la herramienta DDMS con la que, enre otros, es posible simular introducir coordenadas y simular cambios en la localización del dispositivo, entre otras tareas. Todos los datos a emular se deben introducir manualmente, con la consiguiente pérdida de tiempo.

Sin embargo, actualmente resulta imposible emular de algún modo la recepción de señales inalámbricas por parte del dispositivo móvil, algo que de conseguirse, en cualquier caso resultaría un tanto extraño.

6.3. Trabajos futuros

El presente proyecto sienta las bases para un posible futuro desarrollo, actuando como una posible fuente de ideas para proyectos similares como una primera versión de un sistema mejorado. En esta sección se proponen algunas de esas posibles líneas de trabajo:

- Dada la importancia de disponer de un mapa de puntos de acceso lo más fiel posible a la realidad, una posible línea de trabajo supondría investigar e implementar una nueva técnica o un nuevo algoritmo de localización de estos servicios, por ejemplo usando algún tipo de hardware específico, mejorando así la precisión y fiabilidad de la aplicación de recogida de datos. La geolocalización de señales Wi-Fi es un tema puntero y en el cual se está investigando y profundizando actualmente, por lo que no es trivial conseguir buenos resultados.
- La versión actual de la aplicación hace un uso intensivo de la conexión a internet del dispositivo, dado que necesita conectarse al servidor para descargar las posiciones de las redes. Si bien el uso de internet resulta inevitable, sería posible implementar ciertos algoritmos que optimizasen su uso. Por ejemplo, siempre que se descargue la posición de una red pueden descargarse también aquellas posiciones de las redes en un área cercana, pues es de esperar que necesiten ser consultadas próximamente. Otra posible mejora sería la implementación de una base de datos local que actuase como *caché*, asumiendo que existe la posibilidad de que el usuario frecuente habitualmente las mismas zonas y por tanto almacenando éstas de manera permanente en el dispositivo, estando disponibles sin necesidad de ninguna conexión externa.
- Igualmente sería posible replantear la manera en que se usan los datos recogidos por la aplicación Warwalk para después geolocalizar al usuario, si

bien esto implica profundos cambios en todo el sistema. Entre las técnicas propuestas se encuentra el aprendizaje de patrones, de manera que la información recogida no se limite a una única red sino a la totalidad de su entorno en un momento determinado, combinando los datos recogidos para actuar en consecuencia cuando se encuentren situaciones similares en un futuro.

Bibliografía

[1] «*WarDriving. Drive, Detect, Defend.*», por Chris Hurley, Frank Thornton, Michael Puchol & Russ Rogers. Publicado Marzo de 2004. ISBN 978-1-931836-03-6.

[2] Kismet
`http://www.kismetwireless.net/`

[3] KisMAC
`http://trac.kismac-ng.org/`

[4] NetStumbler
`http://www.stumbler.net/`

[5] Wigle.net
`http://www.wigle.net`

[6] Wigle FAQ
`https://www.wigle.net/gps/gps/main/faq/`

[7] Android Market
`http://www.android.com/market/`

[8] The Register, 4 de Marzo de 2010 — «*Apple yanks Wi-Fi detectors from iTunes*»
`http://www.theregister.co.uk/2010/03/04/wifi_stumbling_iphone/`

[9] GigaOM, 4 de Marzo de 2010 — «*Apple Says WiFi-sniffing Apps Stink*»
`http://gigaom.com/2010/03/04/apple-says-wifi-sniffing-apps-stink/`

[10] TechCrunch, 4 de Marzo de 2010 — «*"Problematic Wi-Fi Access": Apple Bans Augmented Reality App Sekai Camera*»
`http://techcrunch.com/2010/03/04/tonchidot-sekai/`

[11] CNET News, 4 de Marzo de 2010 — «*Apple removes Wi-Fi finders from App Store*»
`http://news.cnet.com/8301-13579_3-10464021-37.html`

[12] G-Mon
`http://www.wardriving-forum.de/wiki/G-MoN`

[13] PyNetMony
`http://sites.google.com/site/pynetmony/`

[14] AndroidZoom — WifiTracker
http://www.androidzoom.com/android_applications/tools/wifi-tracker_fyw.html

[15] Ian Hawkins
http://ian-hawkins.com/

[16] AndroidZoom — WigleWiFi
http://www.androidzoom.com/android_applications/tools/wigle-wifi_fxak.html

[17] Skyhook Wireless
http://www.skyhookwireless.com/

[18] Skyhook Wireless — Coverage
http://www.skyhookwireless.com/howitworks/coverage.php

[19] Skyhook Wireless — Company Overview
http://www.skyhookwireless.com/whoweare/

[20] Xconomy Boston, 17 de Enero de 2008 — «*Jobs, iPhone have Skyhook pointed in right direction*»
http://www.xconomy.com/2008/01/17/steve-jobs-sprinkles-a-bit-of-magic-apple-dust-on-bostons-skyhook/

[21] Engadget, 26 de Abril de 2010 — «*Motorola gives Google the boot, turns to Skyhook for location services*»
http://www.engadget.com/2010/04/26/motorola-gives-google-the-boot-turns-to-skyhook-for-location-se/

[22] Skyhook Developers Network
http://www.skyhookwireless.com/developers/

[23] Wordpress, 10 de Septiembre de 2008 — «*Get the physical location of wireless router from its MAC address (BSSID)*»
http://coderrr.wordpress.com/2008/09/10/get-the-physical-location-of-wireless-router-from-its-mac-address-bssid/

[24] Ekahau
http://www.ekahau.com

[25] Official Google Mobile Blog, 28 de Noviembre de 2007 — «*New magical blue circle on your map*»
http://googlemobile.blogspot.com/2007/11/new-magical-blue-circle-on-your-map.html

[26] Google Latitude
http://www.google.com/latitude

[27] Official Google Mobile Blog, 21 de Octubre de 2008 — «*My Location now with Wi-Fi*»
http://googlemobile.blogspot.com/2008/10/my-location-now-with-wi-fi.html

[28] Official Google Blog, 14 de Mayo de 2010 — «*WiFi data collection: An update*»
http://googleblog.blogspot.com/2010/05/wifi-data-collection-update.html

[29] ElectronicsWeekly, 10 de Junio de 2010 — «*Google faces UK criminal complaint over Wi-Fi wardriving*»
http://www.electronicsweekly.com/Articles/2010/06/10/48817/google-faces-uk-criminal-complaint-over-wi-fi-wardriving.htm

[30] TechWorld, 7 de Junio de 2010 — «*Australia to investigate Google WiFi data collection*»
http://news.techworld.com/security/3225919/australia-to-investigate-google-wifi-data-collection/

[31] eWeek, 25 de Junio de 2010 — «*Google audits systems in wake of wardriving scandal*»
http://www.eweekeurope.co.uk/news/scotland-yard-considers-google-wispy-investigation-7996

[32] Google Code — Project Wardrive-Android
http://code.google.com/p/wardrive-android/

[33] Wardrive Database
http://wardrivedb.appspot.com/

[34] The GNU General Public License
http://www.gnu.org/licenses/gpl.html

[35] Traducción al español de la licencia GNU General Public License
http://www.viti.es/gnu/licenses/gpl.html

[36] Wikipedia — dBm
http://en.wikipedia.org/wiki/DBm

[37] OMG Modeling And Metadata Specifications — UML versión 2.3
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

[38] Android
http://www.android.com/

[39] Open Handset Alliance
http://www.openhandsetalliance.com/

[40] Developer Resources for Java Technology
http://java.sun.com/

[41] Android Developers — Application Fundamentals: Application Components
http://developer.android.com/guide/topics/fundamentals.html#appcomp

[42] Android Developers — Application Fundamentals: Component Lifecycles
http://developer.android.com/guide/topics/fundamentals.html#lcycles

[43] Apache Ant
http://ant.apache.org/

[44] OpenStreetMap
http://www.openstreetmap.org/

[45] Google Maps API
http://code.google.com/apis/maps/signup.html

[46] Keytool - Key and Certificate Management Tool
http://java.sun.com/javase/6/docs/technotes/tools/windows/keytool.html

[47] Eclipse.org
http://www.eclipse.org/

[48] Android Developers — Developing in Eclipse, with ADT
http://developer.android.com/guide/developing/eclipse-adt.html

[49] Apache HTTP Server
http://projects.apache.org/projects/http_server.html

[50] MySQL
http://www.mysql.com/

[51] PHPMyAdmin
http://www.phpmyadmin.net/

[52] PHP
http://php.net/

[53] Google Code
http://code.google.com/

[54] Google Code — Proyecto warwalk-android
http://code.google.com/p/warwalk-android/

[55] Google Code — Proyecto wps-android
http://code.google.com/p/wps-android/

[56] LaTeX
http://www.latex-project.org/

[57] Proyecto eInkPlusPlus
http://einkplusplus.uc3m.es/

Glosario

API «Application programming interface», en español interfaz de programación de aplicaciones, es la interfaz o conjunto de funciones implementada por un software, que le permite interactuar con otro software.. 46, 82

BSSID «Basic Service Set IDenfitier» es, en una red inalámbrica, el identificador de un punto de acceso — su dirección MAC — . 15, 27, 65, 66

GPS Sistema de Posicionamiento Global, del inglés *Global Positioning System*.. 9, 13–15, 17, 21, 23, 27, 32, 42, 43, 45, 52, 55, 61–63, 69, 70, 72–75, 78, 80, 85, 86

HTTP «Hypertext Transfer Protocol», o en español protocolo de transferencia de hipertexto, es un protocolo de Internet usado para transacciones entre cliente y servidor.. 40, 42, 83

IDE Siglas en inglés de «*Integrated Development Environment*», en español entorno de desarrollo integrado, es decir un software compuesto por diferentes herramientas de programación para el desarrollo de sistemas.. 51

LPPL Siglas en inglés de «*LaTeX Project Public License*», la Licencia Pública del Proyecto LaTeX.. 83

MAC Del inglés «Media Access Control» o «Control de Acceso al Medio», es un identificador de 48 bits que corresponde de forma única a una tarjeta ethernet o de red.. 15, 18, 21, 22, 27, 35, 37, 65, 66

PDA Asistente Personal Digital, del inglés *Personal Digital Assistant*.. 8, 13

PHP «PHP Hypertext Pre-processor», un lenguaje de programación interpretado diseñado para la creación de páginas web dinámicas.. 36–38, 40, 45

SD «Secure Digital», un popular formato de tarjetas de memoria flash.. 48

SDK «Software Development Kit», siglas en inglés de Kit de Desarrollo de Software.. 21, 22, 24, 46, 51, 52, 58, 69, 82

SMS Siglas en inglés de «*Short Message Service*», o como se le conoce coloquialmente, un mensaje de texto.. 48

SSID «Service Set IDenfitier», es un nombre incluido en todos los paquetes de una red inalámbrica para identificarlos como parte de esa red.. 15, 27, 35, 65

WEP Siglas en inglés de «*Wired Equivalent Privacy*», un estándar obsoleto de seguridad para redes inalámbricas.. 26

WYSIWYG del inglés «*What You See Is What You Get*», en español «Lo que ves es lo que obtienes».. 84

XML Siglas en inglés de «EXtensible Markup Language», en español «Lenguaje de marcas extensible».. 14, 22, 29, 42, 44, 45, 51–55, 58, 64, 67

Anexo A. Instalación de las aplicaciones

En primer lugar, se recuerda que para instalar las aplicaciones Warwalk y WPS es necesario disponer de un dispositivo móvil con sistema operativo Android instalado.

A fecha de la redacción de esta memoria, ninguna de las aplicaciones han sido publicadas en Android Market, por lo que no es posible descargarlas desde el mismo.

En su lugar es posible obtener el archivo instalable `.apk` correspodiente a cada una de ellas desde la página web de los proyectos *Warwalk* [54] y *WPS* [55] en Google Code, así cómo a través de la página web del proyecto eInkPlusPlus [57]. Una vez descargado el archivo `.apk` es necesario copiarlo a la tarjeta de memoria del dispostivo para proceder a su instalación.

Por defecto, Android no permite la instalación de aplicaciones que no provengan de Android Market, por lo que deberá habilitar esta opción a través del menú de ajustes de su dispositivo. En el menú de aplicaciones de su dispositivo Android, seleccione *Ajustes* — *Aplicaciones*, y habilite la opción *Orígenes desconocidos*, tal y como se muestra en la figura 6.2

Figura 6.2: Ajustes de aplicaciones.

Figura 6.1: Página web del proyecto eInkPlusPlus.

Por último, es necesario disponer de alguna de las muchas aplicaciones disponibles en Android Market que permiten instalar archivos `.apk` desde la tarjeta de memoria del dispositivo.

Una de las aplicaciones más sencillas de usar es *Apk Manager*, que puede descargarse gratuitamente desde Android Market. Una vez iniciada, la aplicación buscará de forma automática archivos `.apk` en la tarjeta de memoria del dispositivo, Sólo es necesario seleccionar aquél que se quiera instalar y seguir las instrucciones en pantalla para completar la instalación. La figura 6.3 ilustra el proceso a seguir.

Una vez instalada, la aplicación aparecerá en el menú de aplicaciones de su dispositivo, tal y como se muestra en la figura 6.4.

Figura 6.3: Uso de la aplicación *Apk Manager*.

Figura 6.4: Menú de aplicaciones mostrando *WPS* y *Warwalk*.